МІНІСТЕРСТОВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп‘ютерної та програмної інженерії
Кафедра комп’ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ А.С. Савченко
«___» _____ 20___ р

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СПУПЕНЯ «БАКАЛАВР»

Тема: «Програмна система моніторингу та управління IP адресацією комп‘ютерної мережі»

Виконавець: студент УС-411 Кондрат Максим Сергійович
(студент, група, призвіще, ім'я, по батькові)

Керівник: к. т. н., доцент Савченко Аліна Станіславівна
(науковій ступень, вчене звання, призвіще, ім'я, по батькові)

Нормоконтролер: ст. викл. Шевченко О.П. _____
(П.І.Б.) (підпис)

КИЇВ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп‘ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Освітній ступінь **Бакалавр**

Галузь знань, спеціальність, спеціалізація: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ
Завідувач кафедри

А.С. Савченко
“_____”_____ 2021 р.

ЗАВДАННЯ

на виконання дипломного проекту студента

Кондрата Максима Сергійовича
(прізвище, ім’я, по батькові)

1.Тема проекту: «Програмна система моніторингу та управління IP адресацією комп‘ютерної мережі» затверджена наказом ректора № 636/ст. від 22.04.2021р.

2. Термін виконання роботи: з 10.05.2021 по 20.06.2021р.

3. Вихідні дані до роботи: організація процесу моніторингу та управління IP адресацією комп‘ютерної мережі.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка задачі, огляд технологій та інструментів для розробки системи, розробка функціональності системи та інтерфейсу для користувачів, висновки.

5. Перелік обов’язкового графічного матеріалу: схема етапів розробки системи.

КАЛЕНДАРНИЙ ПЛАН

	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Аналіз літератури та джерел за темою дипломного проекту.	10.05.2021р. – 11.05.2021р.	
2	Розробка та затвердження плану дипломного проекту.	12.05.2021р.	
3	Проведення консультації з науковим керівником щодо створення першого розділ.	13.05.2021р.	
4	Аналітичний огляд і постановка задачі.	14.05.2021р. – 16.05.2021р.	
5	Порівняльний аналіз існуючих систем моніторингу мережі.	17.05.2021р. – 20.05.2021р.	
6	Огляд технологій для розробки системи.	21.06.2021р. – 22.05.2021р.	
7	Розробка компонентів системи.	23.05.2021р. – 03.06.2021р.	
8	Висновки та оформлення пояснювальної записки дипломного проекту.	04.06.2021р. – 08.06.2021р.	
9	Підписання необхідних документів у встановленому порядку.	09.06.2021р. – 10.06.2021р.	
10	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	10.06.2021р. – 11.06.2021р.	

7. Дата видачі завдання: 10.05.2021 р.

Керівник дипломного проекту _____ Савченко А.С.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Кондрат М.С.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Програмна система моніторингу та управління IP адресацією комп'ютерної мережі»: 78 с., 34 рис., 28 літературних джерел.

Об'єкт дослідження: процес адміністрування комп'ютерних мереж.

Предмет дослідження: моніторинг та управління IP адресацією в комп'ютерних мережах.

Мета роботи: розробити систему моніторингу стану мережних вузлів та управління IP адресацією для підвищення ефективності процесу адміністрування комп'ютерної мережі.

Методи дослідження, технічні та програмні засоби: порівняльний аналіз, розробка функціональної складової на мові C#, інтерфейсу користувача за допомогою Windows Forms в середовищі Visual Studio, обробка літературних джерел.

Отримані результати та їх новизна: розроблено систему моніторингу та управління IP адресацією комп'ютерної мережі з інтерфейсом для користувачів та графіками реального часу.

Матеріали дипломного проекту можуть бути використані системними адміністраторами для моніторингу та управління комп'ютерними мережами невеликого масштабу.

МОНІТОРИНГ МЕРЕЖІ, УПРАВЛІННЯ IP АДРЕСАЦІЄЮ, РОБОЧИЙ ПРОЦЕС, C#, WINDOWS FORMS.

ЗМІСТ

ВСТУП 7

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД І ПОСТАНОВКА ЗАДАЧІ 9

1.1. Огляд комп‘ютерних мереж та задач системного адміністрування мереж 10

1.2. Методи моніторингу мережі 18

1.3. Огляд систем моніторингу комп’ютерної мережі 25

1.4. Принципи управління IP-адресацією 31

1.5. Висновки до розділу та постановка задачі 34

РОЗДІЛ 2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ ДЛЯ МОНІТОРИНГУ КОМП’ЮТЕРНОЇ МЕРЕЖІ ТА УПРАВЛІННЯ IP АДРЕСАЦІЄЮ 36

2.1. Технології для розробки функціональної складової системи 37

2.2. Технології для розробки інтерфейсу користувача 39

2.3. Середовище для розробки системи 41

2.4. Висновки до розділу 44

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ МОНІТОРИНГУ КОМП’ЮТЕРНОЇ МЕРЕЖІ ТА УПРАВЛІННЯ IP АДРЕСАЦІЄЮ 46

3.1. Розробка функціональної складової системи 47

3.1.1. Розробка системи моніторингу комп’ютерної мережі 47

3.1.1.1. Створення методів для виявлення мережевих адаптерів комп’ютера .. 48

3.1.1.2. Створення методів для визначення пропускної здатності адаптера 49

3.1.1.3. Створення методів для визначення затримки передачі та частоти втрати пакетів 52

3.1.2. Розробка системи управління IP адресацією 54

3.2. Розробка інтерфейсу користувача 57

3.2.1. Створення інтерфейсу програмування додатків Windows Forms 58
3.2.2. Побудова графіків реального часу 59
3.3. Висновки до розділу 62
ВИСНОВКИ 63
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 65
ДОДАТКИ 68

ВСТУП

Комп'ютерна мережа є сьогодні практично в кожній фірмі та компанії. Мережа вже не вважається чимось розкішним, а є чудовою нагодою ефективно оптимізувати роботу, виробництво, об'єднавши всі комп'ютери в єдину систему [1].

Стежити за роботою всієї мережі, вчасно реагувати на проблеми допомагає система моніторингу мережі. До систем моніторингу мережі відносяться програмні і апаратні засоби, здатні відстежувати різні аспекти мережі і її роботи, такі як трафік, використання смуги пропускання і час безвідмовної роботи. Такі системи можуть виявляти пристрої та інші елементи, які складають мережу або пов'язані з нею, а також забезпечують оновлення статусу.

Метою дипломного проекту є розробка системи моніторингу стану мережних вузлів та управління IP адресацією для підвищення ефективності процесу адміністрування комп‘ютерної мережі.

Об’єктом дослідження даної роботи є процес адміністрування комп’ютерних мереж.

Предмет дослідження: моніторинг та управління IP адресацією в комп‘ютерних мережах.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналітичний огляд предметної області;
- проаналізувати існуючі методи та системи моніторингу;
- проаналізувати принципи та системи управління IP адресацією;
- дослідити та обґрунтувати вибір технології та інструментів для створення системи моніторингу та управління IP адресацією;
- розробити функціональні компоненти та інтерфейс для користувачів системи.

Програмна система моніторингу комп‘ютерної мережі має забезпечувати такий функціонал:

- ідентифікувати мережеві адаптери, доступні на комп'ютері;

- аналізувати пропускну здатність обраного адаптера;
- визначати величину затримки передачі та кількість втрачених пакетів;
- зберігати та візуалізувати зібрані дані (графіки реального часу) для подальшого аналізу.

Програмна система управління IP адресацією комп‘ютерної мережі має забезпечувати такий функціонал: за вказаною IP адресою та маскою:

- визначити адресу мережі/підмережі та хоста;
- визначити широкомовну адресу;
- визначити розмір мережі, кількість хостів і пул допустимих IP адрес;
- зберігати інформацію щодо зайнятих та вільних IP адрес з даного пулу.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД І ПОСТАНОВКА ЗАДАЧІ

У сучасному світі, з високими темпами науково-технічного прогресу, в якому цінність інформації і питома вага інформаційних послуг у всіх сферах суспільного життя різко зросли, виникає потреба в підтримці працездатності локальних мереж. Рішення працездатності локальних мереж вимагає великого штату кваліфікованих фахівців, в обов'язки яких входить підтримка доступності мережевих сервісів, оптимізація роботи мережі, налагодження та оновлення мережевого апаратного і програмного забезпечення, виявлення і усунення виникаючих інцидентів і інші функції. Але замість змісту великого штату IT-співробітників можна використовувати спеціалізоване ПО - системи моніторингу та управління мережею. Основною функцією такого роду програмного забезпечення є моніторинг мережі, тобто постійне спостереження за комп'ютерною мережею в пошуках повільних або несправних систем і оповіщення адміністратора при виявленні збоїв [2].

Моніторинг мережі - це можливість контролювати і убезпечити робочий процес, який пов'язаний з інтернетом і комп'ютерами [3].

Також стає важливим система управління IP-адресами, так як з’являються нові мережі IPv6 з великими адресними пулами 128-бітних шістнадцятирічних чисел і новими методами поділу на підмережі.

Управління IP-адресами - це методологія, реалізована в комп'ютерному програмному забезпеченні для планування і управління призначенням і використанням IP-адрес і тісно пов'язаних ресурсів комп'ютерної мережі. Зазвичай він не надає послуги системи доменних імен (DNS) і протоколу динамічної конфігурації хоста (DHCP), але управляє інформацією для цих компонентів.

1.1. Огляд комп‘ютерних мереж та задач системного адміністрування мереж

Народження комп'ютерних мереж було викликано практичною потребою - мати можливість для спільного використання даних. Персональний комп'ютер - прекрасний інструмент для створення документа, підготовки таблиць, графічних даних та інших видів інформації, але при цьому немає можливості швидко поділитися своєю інформацією з іншими. Коли не було мереж, доводилося роздруковувати кожен документ, щоб інші користувачі могли працювати з ним, або в кращому випадку - копіювати інформацію на дискети. Одночасна обробка документа кількома користувачами виключалася [4].

Об'єднання комп'ютерів у мережі дозволило значно підвищити продуктивність праці. Комп'ютери використовуються як для виробничих (або офісних) потреб, так і для навчання.

Комп'ютерною мережею називають сукупність вузлів (комп'ютерів, терміналів, периферійних пристроїв), що мають можливість інформаційної взаємодії один з одним за допомогою спеціального комунікаційного обладнання та програмного забезпечення.

Розміри мереж варіюються в широких межах - від пари з'єднаних між собою комп'ютерів, що стоять на сусідніх столах, до мільйонів комп'ютерів, розкиданих по всьому світу (частина з них може перебувати на космічних об'єктах).

У мережах застосовуються різні мережеві технології. Кожній технології відповідають свої типи обладнання.

Устаткування мереж підрозділяється на активне і пасивне. Активне обладнання - це інтерфейсні карти комп'ютерів, повторювачі, концентратори; пасивне обладнання - це кабелі, з'єднувальні роз'єми, комутаційні панелі. Крім того, є допоміжне обладнання - пристрої безперебійного живлення, кондиціонування повітря і аксесуари - монтажні стійки, шафи, кабелепроводи різного виду. З точки зору фізики, активне обладнання - це пристрої, яким необхідна подача енергії для генерації сигналів, пасивне устаткування подачі енергії не вимагає.

Устаткування комп'ютерних мереж підрозділяється на кінцеві системи (пристрої), які є джерелами або споживачами інформації, і проміжні системи, щоб забезпечити проходження інформації по мережі.

До кінцевим системам відносять комп'ютери, термінали, мережеві принтери, факс-машини, касові апарати, зчитувачі штрих-кодів, кошти голосового і відео зв’язку і будь-які інші периферійні пристрої.

До проміжних системам відносять концентратори (повторювачі, мости, комутатори), маршрутизатори, модеми та інші телекомунікаційні пристрої, а також з'єднує їх кабельна чи бездротовий інфраструктура.

Для активного комунікаційного обладнання застосовне поняття продуктивність, причому в двох різних аспектах. Крім «валової» кількості неструктурованої інформації, що пропускається обладнанням за одиницю часу (біт/с), важливим є і швидкістю обробки пакетів, кадрів або осередків. Природно, при цьому обумовлюється і розмір структур (пакетів, кадрів, осередків), для якого вимірюється швидкість обробки. В ідеалі продуктивність комунікаційного устаткування повинна бути настільки високою, щоб забезпечувати обробку інформації, що припадає на всі інтерфейси (порти) на їх повній швидкості (wire speed).

Для організації обміну інформацією повинен бути розроблений комплекс програмних і апаратних засобів, розподілених по різним пристроям мережі. Спочатку розробники і постачальники мережевих засобів намагалися йти кожен по своєму шляху, вирішуючи весь комплекс завдань з допомогою власного набору протоколів, програм і апаратури. Однак рішення різних постачальників виявлялися несумісними один з одним, що надавало масу незручностей для користувачів, яких з різних причин не задовольняв набір можливостей, що надаються тільки одним з постачальників. У міру розвитку техніки і розширення асортименту послуг, сервісів назріла необхідність декомпозиції мережевих завдань - роз'єднання їх на кілька взаємопов'язаних підзадач з визначенням правил взаємодії між ними. Розбивка завдання і стандартизація протоколів дозволяє брати участь в її вирішенні великої кількості сторін-розробників програмних і апаратних засобів, виробників

допоміжного і комунікаційного устаткування, доносять всі ці плоди прогресу до кінцевого споживача.

Застосування відкритих технологій і проходження загальновизнаних стандартів дозволяє уникати ефекту вавилонського стовпотворіння.

Всі комп'ютерні мережі можна класифікувати за різними ознаками:

- спосіб організації мережі;
- територіальна поширеність;
- відомча приналежність;
- швидкість передачі інформації;
- тип середовища передачі;
- топологія;
- організація взаємодії комп'ютерів.

За способом організації мережі поділяються на реальні і штучні.

Штучні комп'ютерні мережі (псевдомережі) дозволяють пов'язувати комп'ютери разом через послідовні або паралельні порти і не потребують додаткових пристроях. Іноді зв'язок у такій мережі називають зв'язком по нуль-модему (не використовується модем). Саме з'єднання називають нуль-модемним. Штучні мережі використовуються коли необхідно перекачати інформацію з одного комп'ютера на інший. MS-DOS і Windows забезпечені спеціальними програмами для реалізації нуль-модемного з'єднання. Основним недоліком цих комп'ютерних мереж є низька швидкість передачі даних і можливість з'єднання тільки двох комп'ютерів.

Реальні комп'ютерні мережі дозволяють пов'язувати комп'ютери за допомогою спеціальних пристроїв комутації і фізичної середовище передачі даних. Основним недолік реальних мереж є необхідність в додаткових пристроях.

По територіальній поширеності комп'ютерні мережі поділяються на локальні, глобальні, і регіональні.

Локальні комп'ютерні мережі - це мережі, що перекривають територію не більше 10 кв.м. Вони є мережами закритого типу, доступ до них дозволений тільки обмеженому колу користувачів, для яких робота в такій мережі безпосередньо пов'язана з їх професійною діяльністю.

Регіональні комп'ютерні мережі - це мережі, розташовані на території міста або області.

Глобальні комп'ютерні мережі - це мережі, розташовані на території держави або групи держав. Наприклад, всесвітня мережа Internet. Вони є відкритими і орієнтовані на обслуговування будь-яких користувачів.

Термін «корпоративна мережа» також використовується в літературі і означає об'єднання декількох мереж, кожна з яких може бути побудована на різних технічних, програмних і інформаційних принципах.

За відомчої приналежності розрізняють відомчі і державні мережі. Відомчі комп'ютерні мережі належать одній організації і розташовуються на її території. Державні комп'ютерні мережі - мережі, використовувані в державних структурах.

За швидкістю передачі інформації комп'ютерні мережі поділяються на низько, середньо і високошвидкісні. Низькошвидкісні комп'ютерні мережі - це мережі, що мають швидкість пере-дачі інформації до 10 Мбіт / с. Середньошвидкісні комп'ютерні мережі - це мережі, що мають швидкість передачі інформації до 100 Мбіт / с. Високошвидкісні комп'ютерні мережі - це мережі, що мають швидкість передачі інформації понад 100 Мбіт / с.

За типом середовища передачі комп'ютерні мережі поділяються на коаксіальні, вита пара, оптоволоконні, бездротові (з передачею інформації по радіоканалах, в інфрачервоному діапазоні).

Комп’ютерні мережі можуть класифікуватися також за топологією з'єднання пристроїв. Базовими є топології шинна, кільце, зірка та комбінована.

При використанні шинної топології пристрої мережі з'єднані таким чином, що дані, які передаються між будь-якою парою пристроїв є доступні для всіх інших пристроїв мережі. Визначення, якому саме пристрою та звідки виконується передача, забезпечується шляхом аналізу даних, що передаються всіма пристроями мережі. Надзвичайно проста, але й найменш ефективна топологія. Прикладом мережі з шинною топологією є загально відомий стандарт Ethernet.

Зіркоподібні мережі мають у складі обов'язковий службовий елемент — комутатор. Комутатор не бере участі у комунікаціях безпосередньо, але забезпечує

зв'язок між будь-якою парою пристроїв, які потребують комунікації. Складність та ефективність комутатора значною мірою визначає ефективність всієї мережі.

Характерною рисою розглянутих типів є можливість їх типового застосування на обмеженій території, оскільки зростання довжини комунікаційних каналів призводить до зниження їхньої ефективності та зростання ціни. Наприклад, декілька локальних мереж робочих груп, створених згідно з шинною топологією, можуть бути з'єднані за топологією зорі, а підключення до всесвітньої мережі Internet в найпоширеніших випадках виконується за стільниковою топологією.

З точки зору організації взаємодії комп'ютерів, мережі ділять на однорангові і ієрархічні.

Всі комп'ютери тимчасової мережі рівноправні. Будь-який користувач мережі може отримати доступ до даних, що зберігаються на будь-якому комп'ютері.

Однорангові мережі можуть бути організовані за допомогою таких операційних систем, як Windows'3.11, Novell Netware Lite. Зазначені програми працюють як з DOS, так і з Windows. Однорангові мережі можуть бути організовані також на базі всіх сучасних 32-розрядних операційних систем і деяких інших.

Переваги однорангових мереж:

- найбільш прості в установці і експлуатації;
- операційні системи DOS та Windows володіють усіма необхідними функціями, що дозволяють будувати однорангові з'єднання.

Недолік: в умовах тимчасових мереж ускладнене вирішення питань захисту інформації. Тому такий спосіб організації мережі використовується для мереж з невеликою кількістю комп'ютерів.

В ієрархічній мережі при встановленні мережі заздалегідь виділяються один або кілька комп'ютерів, які керують обміном даних по мережі і розподілом ресурсів. Такий комп'ютер називають сервером. Будь-який комп'ютер, що має доступ до послуг сервера називають клієнтом мережі або робочою станцією.

Сервер в ієрархічних мережах - це постійне сховище поділюваних ресурсів. Сам сервер може бути клієнтом тільки сервера більш високого рівня ієрархії. Тому ієрархічні мережі іноді називаються мережами з виділеним сервером. Сервери

зазвичай являють собою високопродуктивні комп'ютери, можливо, з декількома паралельно працюючими процесорами, з вінчестерами великої місткості, з високошвидкісною мережевою картою (100 Мбіт / с і більше).

Ієрархічна модель мережі є найбільш кращою, так як дозволяє створити найбільш стійку структуру мережі і більш раціонально розподілити ресурси. Також гідністю ієрархічної мережі є більш високий рівень захисту даних.

До недоліків ієрархічної мережі, в порівнянні з однорангових мережами, відносяться:

- необхідність додаткової ОС для сервера;
- більш висока складність установки і модернізації мережі;
- необхідність виділення окремого комп'ютера в якості сервера.

Розрізняють дві технології використання сервера: технологію файл-сервера й архітектуру клієнт-сервер.

У першій моделі використовується файловий сервер, на якому зберігається більшість програм і даних. На вимогу користувача йому пересилаються необхідна програма і дані. Обробка інформації виконується на робочій станції.

У системах з архітектурою клієнт-сервер обмін даними здійснюється між додатком-клієнтом і додатком-сервером. Зберігання даних і їх обробка проводиться на потужному сервері, який виконує також контроль над доступом до ресурсів і даних. Робоча станція отримує тільки результати запиту. Розробники додатків по обробці інформації зазвичай використовують цю технологію.

Причиною, чому необхідно постійно стежити за тим, що відбувається в мережі і швидко розпізнавати всілякі проблеми, є те, що мережеві додатки стали життєво важливими для функціонування підприємств. Ціною проблем з мережею можуть стати для підприємства даремно витрачені робочі години, зруйновані ділові зв'язки і громадський імідж. Тому необхідно володіти інструментами, здатними розпізнавати порушення роботи мережі або доступності мережевих послуг. Управління мережею має за розумну плату забезпечувати контроль мережі, налаштування і аналіз, щоб забезпечити необхідну продуктивність і якість мережевих послуг [5].

Управління мережею (Network management) - цілеспрямований вплив на мережу, здійснюване для організації її функціонування за заданою програмою. Воно включає наступні процедури:

- включення і відключення системи, каналів передачі даних, терміналів;
- діагностика несправностей;
- збір статистики;
- підготовка звітів.

Система управління мережею (Network management system) - апаратні і програмні засоби, що застосовуються для моніторингу та управління вузлами мережі. Програмне забезпечення системи управління мережею складається з агентів, що локалізуються на мережевих пристроях і передають інформацію мережевий керуючої платформі.

Платформа управління мережею (Network management platform) - комплекс програм, призначених для управління мережею і входять в неї системами. Для роботи з платформою адміністратору надається одна або кілька абонентських систем (консолей). Зазвичай платформа створюється на базі протоколу SNMP. Платформа забезпечує:

- контроль роботи пристроїв і стану кабелів;
- контроль ділових процедур;
- контроль інших аспектів функціонування мережі.

Щоб комп'ютерна мережа могла ефективно виконувати свої функції, необхідно централізовано контролювати стан основних її елементів, виявляти і вирішувати виникаючі проблеми, виконувати аналіз продуктивності і планувати розвиток мережі та ін. Ці види робіт є основними завданнями адміністрування мереж.

Основною метою системного адміністрування є приведення мережі у відповідність з цілями і завданнями, для яких вона призначена. Досягається ця мета шляхом управління мережею, що дозволяє мінімізувати витрати часу та ресурсів,

що спрямовуються на управління системою, і в той же час максимізувати доступність, продуктивність і продуктивність системи.

Мережеве адміністрування (Network Management) виникає, коли у адміністратора мережі з'являється потреба і можливість оперувати єдиним представленням мережі, як правило, це відноситься до мереж зі складною архітектурою. При цьому здійснюється перехід від управління функціонуванням окремих пристроїв до аналізу трафіку в окремих ділянках мережі, управління її логічної конфігурацією і конкретними робочими параметрами, причому всі ці операції доцільно виконувати з однієї керуючої консолі. Завдання, які вирішуються в даній області, розбиваються на дві групи: контроль за роботою мережного устаткування та управління функціонуванням мережі в цілому.

Кінцевою метою управління мережею є досягнення параметрів функціонування ІС, які відповідають потребам користувачів. Користувачі оцінюють роботу ІС не по характеристиках мережного трафіку, що застосовуються протоколами, часу відгуку серверів на запити певного типу і особливостей виконуваних сценаріїв управління, а по поводженню додатків, щодня що запускаються на їх настільних комп'ютерах.

Управління мережами здійснюють мережеві адміністратори (адміністратори мереж). Адміністратор мережі - фахівець, який відповідає за нормальне функціонування і використання ресурсів автоматизованої системи та обчислювальної мережі.

Адміністрування інформаційних систем включає наступні цілі:

- установка і налаштування мережі;
- підтримка її подальшої працездатності;
- установка базового програмного забезпечення;
- моніторинг мережі.

У зв'язку з цим адміністратор мережі повинен виконувати такі завдання:

- планування системи;
- установка і конфігурація апаратних пристроїв;

- установка програмного забезпечення;
- установка мережі;
- архівування (резервне копіювання) інформації;
- створення та керування користувачами;
- установка і контроль захисту;
- моніторинг продуктивності.

Забезпечення працездатності системи вимагає і здійснення профілактичних заходів. Адміністратор повинен забезпечувати задоволення санкціонованих запитів користувачів.

1.2. Методи моніторингу мережі

Рішення для моніторингу та діагностики продуктивності мережі отримали активний розвиток у міру прискорення мереж передачі даних і появи нових складних корпоративних додатків і сервісів [6].

Давним-давно мережеві інженери, які обслуговують мережі масштабу підприємства, повинні були просто забезпечувати доступ до мережі і достатню пропускну здатність з'єднання для зв'язку з різними серверами, а також коректної роботи додатків і кінцевих пристроїв. З точки зору мережевої моделі OSI, основний акцент їх роботи був спрямований на рівні 1-4. Верхні рівні моделі OSI в більшій чи меншій мірі ігнорувалися, так як потоки даних і весь трафік передавалися по мережі через загальні смуги пропускання і черги ресурсів.

Час минав, і мережеве обладнання ставало все складніше, дозволивши по-різному, аж до конкретної точки мережі, ідентифікувати і обробляти різні потоки даних. Для цих цілей використовувалися різні моделі якості обслуговування (quality of service, QoS) і методи формування трафіку (traffic shaping) в залежності від пріоритетів додатків. Крім того, все зростаюча залежність від критично важливих для бізнесу додатків змусила мережевих інженерів розбиратися з верхніми рівнями мережевої моделі OSI, щоб вони могли бути в змозі допомогти ідентифікувати будь-які інциденти або проблеми, пов'язані з мережею, серверної операційної системою,

програмним забезпеченням для віртуалізації і самими додатками. Але для того, щоб це зробити, необхідний інструментарій, щоб мати можливість ідентифікувати подібні проблеми і коректно їх усунути.

Здебільшого рішення для моніторингу та діагностики продуктивності мережі розвинулися з більш традиційного і менш складного програмного забезпечення для моніторингу мережі. Ці інструменти моніторингу для отримання інформації про стан мережі зазвичай використовують утиліту Ping, що працює на базі повідомлень протоколу ICMP (Internet Control Message Protocol, протокол міжмережевих керуючих повідомлень), що входить в стек протоколів TCP/IP, а також можливості щодо забезпечення синхронізації та проведення опитувань з центру моніторингу (комбінація polling/traps) на основі протоколу SNMP (Simple Network Management Protocol, простий протокол мережевого управління). Більш сучасні реалізації включають в себе можливості моніторингу, а також візуального представлення базового та інтелектуального аналізу стану всієї мережі аж до самих додатків. Більшість сучасних інструментів для моніторингу продуктивності мережі дозволяють виконувати п'ять наступних функціональних можливостей:

- моніторинг мережі і додатків;
- виявлення проблем з віртуалізацією і операційними системами;
- аналіз мережевих проблем;
- аналіз захоплених даних додатків і потоків;
- пошук кореневої причини інциденту або проблеми.

Вибір способів і об'єктів моніторингу мережі залежить від багатьох факторів - зміни мережі, що діють в ній сервісів і служб, конфігурації серверів і встановленого на них ПЗ, можливостей забезпечення, яке використовується для моніторингу [7].

Початковий рівень будь-якої перевірки - тестування фізичної доступності обладнання (яке може бути порушене в результаті відключення самого обладнання або відмову каналів зв'язку). Як мінімум, це означає перевірку доступності по ICMP-протоколу (ping), причому бажано перевіряти не тільки факт наявності відповіді, але і час проходження сигналу, і кількість втрачених запитів: аномальні значення цих величин, як правило, сигналізують про серйозні проблеми в конфігурації мережі .

Деякі з цих проблем легко відстежити за допомогою трасування маршруту (traceroute) - її також можна автоматизувати за наявності «еталонних маршрутів».

Наступний етап - перевірка принципової працездатності критичних служб. Як правило, це означає TCP-підключення до відповідного порту сервера, на якому повинна бути запущена служба, і, можливо, виконання тестового запиту (наприклад, аутентифікації на поштовому сервері за протоколом SMTP або POP або запит тестової сторінки з веб-сервера).

У більшості випадків, бажано перевіряти не тільки факт відповіді служби / сервісу, але і затримки - втім, то відноситься вже до наступної за важливістю завдання: перевірці навантаження. Крім часу відгуку пристроїв і служб для різних типів серверів існують інші принципово важливі перевірки: пам'ять і завантаженість процесора (веб-сервер, сервер БД), місце на диску (файл-сервер), і більш специфічні - наприклад, статус принтерів у сервера друку.

Способи перевірки цих величин варіюються, але один з основних, доступних в більшості випадків - перевірка по SNMP-протоколу. Крім цього, можна використовувати специфічні засоби, що надаються ОС перевіряється обладнання: наприклад, сучасні серверні версії ОС Windows на системному рівні надають так звані лічильники продуктивності (performance counters), з яких можна "прочитати" досить докладну інформацію про стан комп'ютера.

Розглянемо два способи моніторингу мережі: перший - маршрутизатор-орієнтований, другий - не орієнтований на маршрутизатори. Функціональність моніторингу, який вбудований в самі маршрутизатори і не вимагає додаткової установки програмного або апаратного забезпечення, називають методами, заснованими на маршрутизаторі. Які базуються на маршрутизаторах методи вимагають установки апаратного і програмного забезпечення і надають велику гнучкість [8].

Методи моніторингу засновані на маршрутизаторі - вшиті в маршрутизаторах і, отже, мають низьку гнучкість. Одним з найбільш часто використовуваного метода такого моніторингу є протокол простого мережевого моніторингу (SNMP). SNMP - протокол прикладного рівня, який є частиною протоколу TCP/IP. Він дозволяє

адміністраторам керувати продуктивністю мережі, знаходити й усувати проблеми, планувати зростання мережі. Він збирає статистику по трафіку до кінцевого хоста через пасивні датчики, які реалізуються разом з маршрутизатором. Для протоколу SNMP притаманні три ключові компоненти: керовані пристрої (Managed Devices), агенти (Agents) та системи управління мережею (Network Management Systems - NMSs).

Керовані пристрої включають в себе SNMP-агента і можуть складатися з маршрутизаторів, перемикачів, комутаторів, концентраторів, персональних комп'ютерів, принтерів і інших елементів, подібних до цих. Вони несуть відповідальність за збір інформації та роблять її доступною для системи управління мережею (NMS).

Агенти включають в себе програмне забезпечення, яке володіє інформацією з управління, і переводять цю інформацію в форму, сумісну з SNMP. Вони закриті для пристрою управління.

Системи управління мережею (NMS) виконують додатки, які займаються моніторингом і контролем пристроїв управління. Ресурси процесора і пам'яті, які необхідні для управління мережею, надаються NMS. Для будь-якої керованої мережі повинна бути створена хоча б одна система управління. SNMP може діяти виключно як NMS, або як агент.

Існує 4 основних команди, що використовуються SNMP NMS для моніторингу і контролю керованих пристроїв: читання, запис, переривання і операції перетину. Операція читання розглядає змінні, які зберігаються керованими пристроями. Команда записи змінює значення змінних, які зберігаються керованими пристроями. Операції перетину володіють інформацією про те, які змінні керованих пристроїв підтримують, і збирають інформацію з підтримуваних таблиць змінних. Операція переривання використовується керованими пристроями для того, щоб повідомити NMS про настання певних подій.

Як говорилося раніше, SNMP - протокол рівня додатків, який використовує пасивні сенсори, щоб допомогти адміністратору простежити за мережевим трафіком і продуктивністю мережі. Хоча, SNMP може бути корисним інструментом для

мережевого адміністратора, він також створює можливість для загрози безпеки, тому що він позбавлений можливості аутентифікації.

Хоча технології, які не вбудовані в маршрутизатор все ж обмежені в своїх можливостях, вони пропонують велику гнучкість, ніж технології вбудовані в маршрутизатори. Ці методи класифікуються як активні і пасивні.

Активний моніторинг повідомляє проблеми в мережі, збираючи вимірювання між двома кінцевими точками. Система активного виміру має справу з такими метриками, як: корисність, маршрутизатори/маршрути, затримка пакетів, повтор пакетів, втрати пакетів, нестійка синхронізація між прибуттям, вимір пропускної здатності.

Головним чином використання інструментів, такі як команда ping, яка вимірює затримку і втрати пакетів, і traceroute, яка допомагає визначити топологію мережі, є прикладом основних активних інструментів вимірювання. Обидва ці інструменту посилають пробні ICMP-пакети до точки призначення і чекають, коли ця точка відповість відправнику.

Даний метод може не тільки збирати одиничні метрики про активний вимірі, а й може визначати топологію мережі. Ще один важливий приклад активного виміру - утиліта iperf. Iperf - це утиліта, яка вимірює якість пропускної здатності TCP і UDP протоколів. Вона повідомляє пропускну здатність каналу, існуючу затримку і втрати пакетів.

Проблема, яка існує з активним моніторингом, - це те, що представлені проби в мережі можуть втручатися в нормальний трафік. Часто час активних проб обробляється інакше, ніж нормальний трафік, що ставить під питання важливість наданої інформації від цих проб.

Відповідно до загальної інформації, описаної вище, активний моніторинг - це надзвичайно рідкісний метод моніторингу, взятий окремо. Пасивний моніторинг навпаки не вимагає великих мережевих витрат.

Пасивний моніторинг на відміну від активності не додає трафік в мережу і не змінює трафік, який вже існує в мережі. Також на відміну від активного моніторингу, пасивний збирає інформацію тільки про одну точці в мережі.

Вимірювання відбуваються набагато краще, ніж між двома точками, при активному моніторингу.

Пасивні вимірювання мають справу з такою інформацією, як: трафік і суміш протоколів, кількість бітів (швидкість), синхронізація пакетів і час між прибуттям. Пасивний моніторинг може бути здійснений, за допомогою будь-якої програми, витягаючої пакети.

Хоча пасивний моніторинг не має витрат, які має активний моніторинг, він має свої недоліки. З пасивним моніторингом, вимірювання можуть бути проаналізовані тільки оф-лайн і вони не представляють колекцію. Це створює проблему, пов'язану з обробкою великих наборів даних, які зібрані під час вимірювання.

Пасивний моніторинг може бути краще активного в тому, що дані службових сигналів не додаються в мережу, але пост-обробка може викликати велику кількість тимчасових витрат. Ось чому існує комбінація цих двох методів моніторингу.

Отже можна переходити до висновку про те, що комбінування активного і пасивного моніторингу - кращий спосіб, ніж використання першого або другого окремо. Комбіновані технології використовують кращі сторони і пасивного, і активного моніторингу середовищ. Дві нові технології, що представляють комбіновані технології моніторингу, описуються нижче. Це «Перегляд ресурсів на кінцях мережі» (WREN) і «Монітор мережі з власною конфігурацією» (SCNM).

WREN використовує комбінацію технік активного і пасивного моніторингу, активно обробляючи дані, коли трафік малий, і пасивно обробляючи дані протягом часу великого трафіку. Він дивиться трафік і від джерела, і від одержувача, що робить можливим більш точні вимірювання. WREN використовує трасування пакетів від створеного додатком трафіку для вимірювання корисної пропускної здатності. WREN розбитий на два рівня: основний рівень швидкої обробки пакетів і аналізатор трасування призначеного для користувацького рівня.

Основний рівень швидкої обробки пакетів відповідає за отримання інформації, пов'язаної з вхідними та вихідними пакетами. На рис.1.1 показує список інформації, яка збирається для кожного пакета. Доступ до буферу здійснюється за допомогою

двох системних викликів. Один виклик починає трасування і надає необхідну інформацію для її збору, поки другий виклик повертає трасування з ядра.

Incoming Packets				Outgoing Packets			
timestamp	seq #	ack #	TCP cwnd	timestamp	seq #	ack #	data size

Рис.1.1. Інформація, зібрана на головному рівні трасування пакетів

Аналізатор трасування призначеного для користувацького рівня - інший рівень в середовищі WREN. Це компонент, який починає трасування будь-якого пакета, збирає і обробляє повернуті дані на рівні ядра оператора. Згідно проектування, компоненти користувальницького рівня не мають потреби в читанні інформації від об'єкта трасування пакетів весь час. Вони можуть бути проаналізовані негайно після того, як трасування буде завершена, щоб зробити висновок в реальному часі, або дані можуть бути збережені для подальшого аналізу.

Загалом, WREN - це дуже корисна установка, яка використовує переваги і активного, і пасивного моніторингу. Хоча ця технологія знаходиться на ранньому етапі розвитку, WREN може надати адміністраторам корисні ресурси в моніторингу та аналізу їх мереж.

Монітор з власною конфігурацією мережі (SCNM) - інший інструментарій, який використовує технології та активного, і пасивного моніторингу. SCNM - це інструмент моніторингу, який використовує зв'язок пасивних і активних вимірювань для збору інформації на 3 рівні проникнення, що виходять маршрутизаторів, і інших важливих точок моніторингу мережі. Середа SCNM включає і апаратний, і програмний компонент.

Апаратний засіб встановлюється в критичних точках мережі. Він відповідає за пасивний збір заголовків пакетів. Програмне забезпечення запускається на кінцевій точці мережі.

Програмне забезпечення відповідає за створення і посилку активованих пакетів, які використовуються для старту моніторингу мережі. Користувачі будуть

посилати в мережу пакети активації, що містять деталі про пакетах, які вони хочуть отримати для моніторингу та збору. Його користувачі не потребують знання місця розташування SCNM-хоста, приймаючи за істину те, що всі хости відкриті для «прослушки» пакетів. На основі інформації, яка існує в рамках активованого пакета, фільтр поміщається в потік збору даних, який також працює в кінцевій точці. Збираються заголовки пакетів мережевого і транспортного рівня, які відповідають фільтру. Фільтр буде автоматично введений в тайм аут. Служба вибірки пакетів, яка запускається на SCNM-хості, використовує команду tcpdump (подібно програмі вибірки пакетів) в порядку отриманих запитів і записи трафіку, який відповідає запиту.

Коли інструментами пасивного моніторингу визначається проблема, трафік може бути згенерований за допомогою інструментів активного моніторингу, дозволяючи збирати додаткові дані для більш детального вивчення проблеми. При розгортанні цього монітора в мережі на кожному маршрутизаторі на протязі шляху, ми може вивчати тільки секції мережі, які мають проблеми.

SCNM призначений для установки і використання, головним чином, адміністраторами. Проте звичайні користувачі можуть використовувати деяку частину цієї функціональності. Хоча звичайні користувачі здатні використовувати частини середовища SCNM моніторингу, їм дозволено дивитися тільки свої власні дані.

Отже, SCNM - це ще один спосіб комбінованого моніторингу, який використовує і активний, і пасивний методи, щоб допомогти адміністраторам моніторити і аналізувати їх мережі.

1.3. Огляд систем моніторингу комп’ютерної мережі

Автоматизовані системи моніторингу грають важливу роль в житті сучасного суспільства. Моніторинг – це безперервний процес спостереження і реєстрації параметрів об'єкта в порівнянні з заданими критеріями. У ряді галузей дані збираються і накопичуються дуже інтенсивно [9].

Через тенденції зростання мереж передачі даних підвищуються вимоги до систем моніторингу цих мереж. При розширенні мережі збільшується і кількість складових її об'єктів, які потребують моніторингу, а отже, і навантаження на систему моніторингу. Така ситуація спричиняє за собою уповільнення реакції на аварійні події в мережі, веде до деградації мережевих сервісів і служб. Для запобігання цим ситуаціям необхідний правильний підхід до вибору системи моніторингу. Нижче розглядається декілька найбільш популярних систем моніторингу: Cacti, Nagios, Zabbix.

Cacti - це безкоштовна програма, що входить в LAMP-набір (Linux + Apache + MySQL + PHP/Perl/Python) серверного програмного забезпечення, яке надає стандартизовану програмну платформу для побудови графіків на основі практично будь-яких статистичних даних (рис. 1.2). Якщо будь-який пристрій або сервіс повертає числові дані, то вони, швидше за все, можуть бути інтегровані в Cacti. Існують шаблони для моніторингу широкого спектру обладнання - від Linux- і Windows-серверів до маршрутизаторів і комутаторів Cisco, - в основному все, що спілкується на SNMP (Simple Network Management Protocol, простий протокол мережевого управління). Існують також колекції шаблонів від сторонніх розробників, які ще більше розширюють і без того величезний список сумісних з Cacti апаратних засобів і програмного забезпечення [10].

Незважаючи на те, що стандартним методом збору даних Cacti є протокол SNMP, також для цього можуть бути використані сценарії на Perl або PHP. Фреймворк програмної системи вміло розділяє на дискретні екземпляри збір даних і їх графічне відображення, що дозволяє з легкістю повторно обробляти і реорганізовувати існуючі дані для різних візуальних уявлень. Крім того, ви можете вибрати певні часові рамки і окремі частини графіків просто клікнувши на них і перетягнувши.

Таким чином, Cacti - це інструментарій з великими можливостями для графічного відображення та аналізу тенденцій продуктивності мережі, який можна використовувати для моніторингу практично будь-який контрольованої метрики, що подається у вигляді графіка. Дане рішення також підтримує практично безмежні

можливості для налаштування, що може зробити його занадто складним при певних застосуваннях.

Недоліки Cacti:

- складний в початковому налаштуванні;
- не ефективні в моніторингу SNMP;
- не пропонують стільки розширень та плагінів, скільки його аналоги;
- не підтримує реляційну базу даних для зберігання інформації;
- використовується тільки для візуалізації [12].

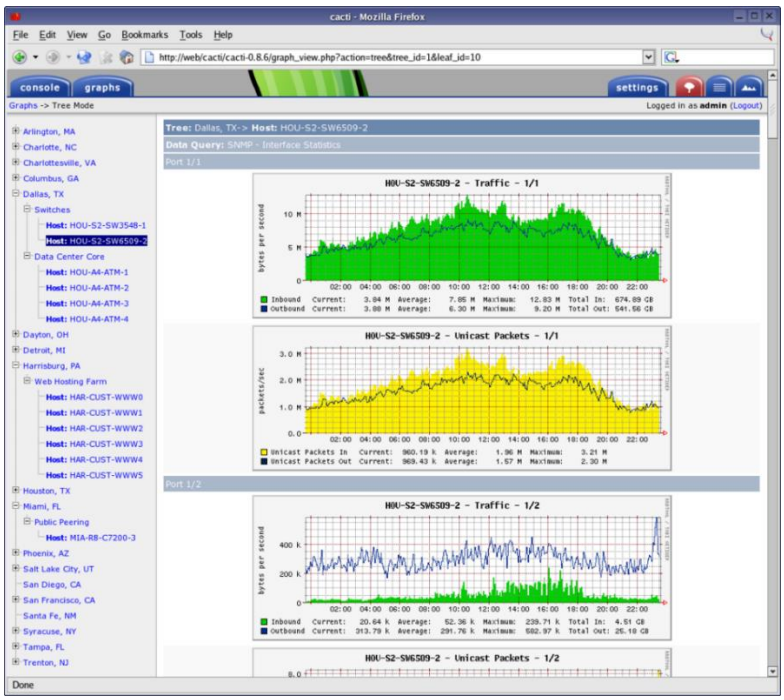

Рис. 1.2. Приклад графіків в Cacti

Nagios - це програмна система для моніторингу мережі, яка вже багато років знаходиться в активній розробці (рис. 1.3). Написана на мові C, вона дозволяє робити майже все, що може знадобиться системним і мережевим адміністраторам від пакета прикладних програм для моніторингу. Веб-інтерфейс цієї програми є швидким та інтуїтивно зрозумілим, в той час його серверна частина - надзвичайно надійною [10].

Як і Cacti, дуже активна спільнота підтримує Nagios, тому різні плагіни існують для величезної кількості апаратних засобів і програмного забезпечення. Від

найпростіших ping-перевірок до інтеграції зі складними програмними рішеннями, такими як, наприклад, написаним на Perl безкоштовним програмним інструментарієм WebInject для тестування веб-додатків і веб сервісів. Nagios дозволяє здійснювати постійний моніторинг стану серверів, сервісів, мережевих каналів і всього іншого, що розуміє протокол мережевого рівня IP. Наприклад, ви можете контролювати використання дискового простору на сервері, завантаженість ОЗУ і ЦП, використання ліцензії FLEXlm, температуру повітря на виході сервера, затримки в WAN і Інтеренет-каналі і багато іншого.

Очевидно, що будь-яка система моніторингу серверів і мережі не буде повноцінною без повідомлень. У Nagios програмна платформа пропонує налаштувати механізм повідомлень по електронній пошті, через СМС та миттєві повідомлення більшості популярних Інтернет-месенджерів, а також схему ескалації, яка може бути використана для прийняття розумних рішень про те, хто, як і при яких обставин повинен бути повідомлений, що при правильному налаштуванні допоможе вам забезпечити багато годин спокійного сну. А веб-інтерфейс може бути використаний для тимчасового призупинення отримання повідомлень або підтвердження трапилася проблеми, а також внесення заміток адміністраторами.

Недоліки Nagios:

- Немає можливості конфігурації через web-інтерфейс (для безкоштовної версії). Всі зміни конфігурації виконуються правкою файлів конфігурації з подальшим повним перезапуском сервера Nagios (~ 10-15 хвилин);
- Занадто великий інтервал між перевірками і вимірами параметрів;
- RRD усереднює дані, тому неможливо сказати, яке було точне значення параметрів, наприклад, місяць тому;
- Відсутні вбудовані засоби візуалізації;
- Кожен плагін запускається як окремий процес [11].

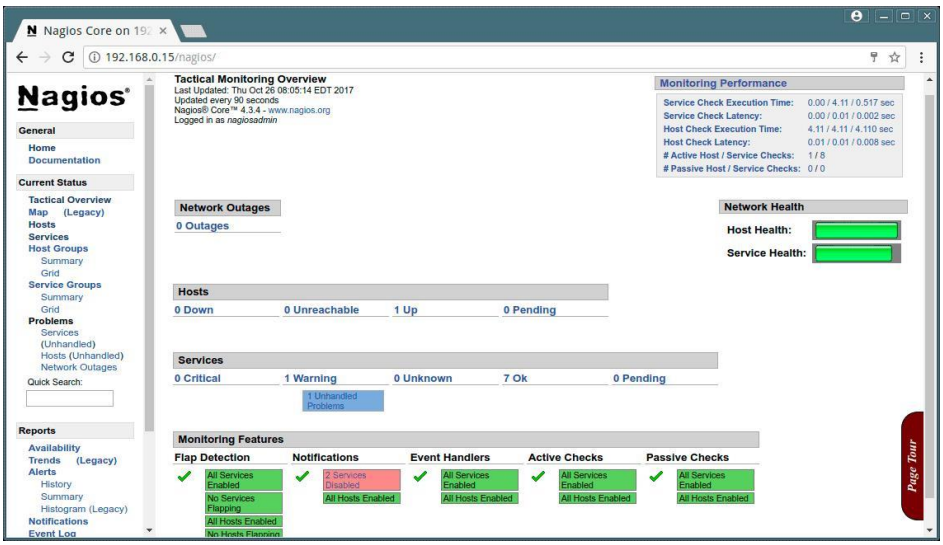

Рис. 1.3. Інтерфейс Nagios

Zabbix - це повномасштабний інструмент для мережевого і системного моніторингу мережі, який об'єднує декілька функцій в одній веб-консолі (рис. 1.4). Він може бути налаштований для моніторингу та збору даних з різних серверів і мережевих пристроїв, забезпечуючи обслуговування і моніторинг продуктивності кожного об'єкта [10].

В основному, Zabbix працює з програмними агентами, запущеними на контрольованих системах. Але це рішення також може працювати і без агентів, використовуючи протокол SNMP або інші можливості для здійснення моніторингу. Особлива увага також приділяється моніторингу серверів додатків Java, веб-сервісів і баз даних.

Хости можуть додаватися вручну або через процес автоматичного виявлення. Широкий набір шаблонів за замовчуванням застосовується до найбільш поширеним варіантам використання, таким як Linux, FreeBSD і Windows-сервера; широко-використовувані служби, такі як SMTP і HTTP, а також ICMP і IPMI для докладного моніторингу апаратної частини мережі. Крім того, призначені для користувача перевірки, написані на Perl, Python або майже на будь-якій іншій мові, можуть бути інтегровані в Zabbix.

Zabbix дозволяє налаштовувати панелі моніторингу та веб-інтерфейс, щоб сфокусувати увагу на найбільш важливих компонентах мережі. Відомості та

ескалації проблем можуть ґрунтуватися на налаштовуваних діях, які застосовуються до хостів або груп хостів. Дії можуть навіть налаштовуватися для запуску віддалених команд, отже якийсь ваш сценарій може запускатися на контрольованому хості, якщо спостерігаються певні критерії подій.

Програма відображає у вигляді графіків дані про продуктивність, такі як пропускна здатність мережі та завантаження процесора, а також збирає їх для настроюються систем відображення. Крім того, Zabbix підтримує карти, екрани і навіть слайд-шоу, що відображають поточний статус контрольованих пристроїв.

Недоліки Zabbix:

- моніторинг серверів і робочих станцій здійснюється через постійно запущений агент;
- всі дані моніторингу зберігаються в базі, що в великих мережах вимагає виділення додаткових обчислювальних потужностей для обслуговування бази даних;
- складність реалізації на початковому етапі;
- не забезпечується відмовостійкість [11].

Рис. 1.4. Інтерфейс Zabbix

1.4. Принципи управління IP-адресацією

Схема IP-адресації, що застосовується в TCP / IP, дозволяє користувачам і додаткам однозначно ідентифікувати мережі і хости, з якими встановлюються з'єднання. IP-адреса працює так само, як і поштова адреса, дозволяючи направляти дані в обраний пункт призначення. Протокол TCP/IP описує стандарти для присвоєння адрес мережам, підмережам, хостам та сокетам [13].

IP-адреса — це ідентифікатор (унікальний числовий номер) мережевого рівня, який використовується для адресації комп'ютерів чи пристроїв у мережах, які побудовані з використанням протоколу TCP/IP.

У протоколі Internet використовуються IP-адреси довжиною 32 розряду, що складаються з двох частин. 32 розряди розділені на чотири октети - 01111101 00001101 01001001 00001111.

Значення октетів зазвичай записуються в десятковій нотації - 125 13 73 15.

IP-адреса складається з адреси мережі і адреси хоста (або локальної адреси). Така адреса, що складається з двох частин, дозволяє відправнику задавати як мережу, так і конкретний хост в цій мережі. Кожній мережі присвоюється унікальна адреса при приєднанні її до інших мереж Internet. Однак, якщо ви не плануєте підключати локальну мережу до інших мереж Internet, їй можна присвоїти будь-яку мережеву адресу.

Розподіл на адреси мережі та вузла відбувається за допомогою маски - 4-х байтного числа, яке поставлено у відповідність IP-адресою. Макса містить 1 в тих розрядах IP-адреси, які визначають адресу мережі, і 0 в тих розрядах IP адреси, які визначають адреса вузла [14].

Адресою IP-мережі вважається IP-адреса з цієї мережі, в якому в поле адреси вузла містяться 0. Ця електронна адреса позначає мережу цілком в таблицях маршрутизації. Є ще службова IP-адреса - широкомовна адреса або броадкаст, в поле адреси вузла вона містить 1. Обидва ці адреси не використовуються для адресації реальних вузлів мережі, однак входять в діапазон адрес IP-мережі.

В TCP / IP передбачено три класи IP-адрес: A, B і C. Від класу залежить довжина адреси мережі (і адреси хоста) в IP-адресу. Залежно від розміру мережі, вона може бути віднесена до того чи іншого класу [13].

Адреса класу A складається з 8-розрядної адреси мережі і 24-розрядної локального адреси або адреси хоста. Перший біт в мережевому адресу призначений для вказівки класу мережі, решта 7 бітів - реальну адресу мережі. Оскільки максимальне число, яке можна представити в двійковому вигляді семи бітами, дорівнює 128, то в класі A може бути 128 адрес мережі. Два адреси з цих 128 можливих адрес зарезервовані для спеціальних випадків: мережеву адресу 127 зарезервований для локальних циклічних адрес, а мережеву адресу, що складається з одних одиниць, означає адресу оповіщення. Таким чином, існує 126 можливих адрес мережі класу A і 16 777 214 адрес локальних хостів. В адресі класу A старший розряд дорівнює 0.

Адреса класу B складається з 16-розрядної адреси мережі і 16-розрядної адреси хоста. Перші два біта в адресі сіті призначені для вказівки класу мережі, решта 14 бітів - реальну адресу мережі. Існує 16 384 можливих адрес мережі та 65 536 адрес локальних хостів. В адресі класу B два старших розряду рівні 10. Значення першого октету адрес класу B лежать в діапазоні від 128 до 191.

Адреса класу C складається з 24-розрядної адреси мережі і 8-розрядної адреси локального хоста. Перші два біта в адресі призначені для вказівки класу мережі, решта 22 біта - реальну адресу мережі. Таким чином, існує 2 097 152 можливих адреси мережі і 256 адрес локальних хостів. В адресі класу C два старших розряду рівні 11. Іншими словами, значення першого октету адрес класу C лежать в діапазонах від 192 до 223.

Також існує 2 додаткові класи - D та E. Клас D - використовується для багатоадресної розсилки. Перші октети IP-адрес класа D знаходяться в діапазоні від 224 до 239. Клас E - зарезервований для експериментального використання і містить діапазон адрес, в яких перший октет розташований в діапазоні від 240 до 255 [15].

Також існує зворотня маска, вона відрізняється від прямої маски тим, що зворотня маска не повинна містити поспіль розташовані одиниці або нулі, і одиниці це не просто позначення області хоста в IP-адресі. Одиниці це позначення бітів в IP-адресі, які можуть змінюватися під час перевірки умов, а нулі фіксують незмінні біти [16].

Область застосування зворотньої маски це умовні операції з IP-адресами. До цих областей відносяться зокрема списки доступу (ACL) - зворотня маска дозволяє створювати не просто умови хоста з цієї мережі, а набагато більш гнучкі правила і визначення мереж, а також конфігурація протоколів маршрутизації.

Отже, знаючи IP-адресу та маску підмережі ми можемо визначити адресу мережі, широкомовну адресу, зворотню маску, кількість та діапазон хостів в мережі.

Щоб визначити адресу мережі та широкомовну адресу потрібно представити IP-адресу та маску підмережі у двійковому вигляді. Наприклад, IP-адреса - 169.254.43.226 та маска підмережі - 255.255.224.0. В двійковому вигляді це буде виглядати так:

169.254.43.226 - 10101001.11111110.001 01011.11100010
255.255.224.0 - 11111111.11111111.111 00000.00000000

Адрес мережі буде дорівнювати 10101001.11111110.001 00000.00000000, тобто 169.254.32.0.

Широкомовна адреса - 10101001.11111110.001 11111.11111111, тобто 169.254.63.255.

Зворотня маска підмережі 00000000.00000000.000 11111.11111111 - 0.0.31.255.

Знаючи адресу мережі та широкомовну адресу можна визначити діапазон хостів - від 169.254.32.1 до 169.254.63.254. Далі можемо визначити кількість доступних хостів в мережі за формулою (2^n)-2, де n – розмір ідентифікатора хоста, в даному випадку від дорівнює 13, а, отже, кількість хостів - 8190.

1.5. Висновки до розділу та постановка задачі

Постійний контроль за роботою локальної мережі, необхідний для підтримки її в працездатному стані. Контроль - це необхідний етап, який повинен виконуватися при управлінні мережею. Процес контролю роботи мережі зазвичай ділять на два етапи - моніторинг і аналіз.

На етапі моніторингу виконується процедура збору первинних даних про роботу мережі: статистика про кількість циркулюючих в мережі кадрів і пакетів різних протоколів, стан портів концентраторів, комутаторів і маршрутизаторів.

Далі виконується етап аналізу, під яким розуміється процес осмислення зібраної на етапі моніторингу інформації, зіставлення її з даними, отриманими раніше, і вироблення припущень про можливі причини сповільненої або ненадійної роботи мережі.

На сьогодні існує багато систем моніторингу та управління IP адресацією комп‘ютерної мережі, але вони мають деякі недоліки:

- надмірна функціональність;
- необхідність стабільної оплати (переважно високої) за використання сервісів та додатків;
- складність використання.

Отже, виникла необхідність створення власної системи для уникнення зазначених проблем.

Враховуючи вищенаведене, задача розробки системи моніторингу стану мережних вузлів та управління IP адресацією для підвищення ефективності процесу адміністрування комп‘ютерної мережі є актуальною. Особливо важливим це завдання є для організацій, які мають мережі з невеликою кількістю пристроїв. Метою дипломного проекту є розробка системи моніторингу та управління IP адресацією комп‘ютерної мережі, яка має забезпечувати такий функціонал:

- ідентифікувати мережеві адаптери, доступні на комп'ютері;
- аналізувати пропускну здатність обраного адаптера;
- визначати величину затримки передачі та кількість втрачених пакетів;

- зберігати та візуалізувати зібрані дані (графіки реального часу) для подальшого аналізу.

Система управління IP адресацією комп‘ютерної мережі має забезпечувати такий функціонал: за вказаною IP адресою та маскою:

- визначити адресу мережі;
- визначити широкомовну адресу;
- визначити розмір мережі, кількість хостів і діапазон допустимих IP адрес в мережі;
- зберігати інформацію щодо зайнятих IP адрес в даній мережі.

Перевагами системи, що пропонується, є безкоштовне використання, легкість в розумінні та невеликий функціонал, що показує основну інформацію про мережу.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ ДЛЯ МОНІТОРИНГУ КОМП'ЮТЕРНОЇ МЕРЕЖІ ТА УПРАВЛІННЯ IP АДРЕСАЦІЄЮ

Як правило, в додатку визначаються рівні призначені для інтерфейсу користувача, бізнес-логіки і доступу до даних (рис. 2.1). В рамках такої архітектури користувачі виконують запити через інтерфейс користувача, який взаємодіє тільки з рівнем бізнес-логіки. Рівень бізнес-логіки, в свою чергу, може викликати шар доступу до даних для обробки запитів. Рівень інтерфейсу користувача не повинен виконувати запити безпосередньо до рівня доступу до даних і будь-якими іншими способами безпосередньо взаємодіяти з функціями зберігання. Для кожного рівня чітко визначені свої обов'язки.

Рис. 2.1. Архітектура типового додатка

Одним з недоліків даного підходу є те, що обробка залежностей під час компіляції здійснюється зверху вниз. Це означає, що рівень інтерфейсу користувача залежить від рівня бізнес-логіки, який, в свою чергу, залежить від рівня доступу до даних. Це означає, що рівень бізнес-логіки, який зазвичай містить ключові функції програми, залежить від деталей реалізації доступу до даних.

Система моніторингу та управління IP адресацією комп‘ютерної мережі включає технології для розробки функціональної складової системи (рівні бізнес-логіки і доступу до даних) та інтерфейсу користувача.

2.1. Технології для розробки функціональної складової системи

Основна функціональна складова системи моніторингу та управління IP адресацією комп‘ютерної мережі написана на мові C#. C# - сучасна об'єктно- і компонентно-орієнтована мова програмування. C# дозволяє розробникам створювати безліч безпечних і надійних програм, які працюють на фреймворці .NET [18].

.NET - це фреймворк від Microsoft, який дозволяє використовувати одні і ті ж простори імен, бібліотеки і API для різних мов. .Net Framework - це набір вже скомпільованих бібліотек, звідки беруться методи і функції для запуску і розробки додатків. Платформа .NET не прив'язана до операційної системи Windows і згодом може бути реалізована для інших операційних систем. .NET - це безкоштовна платформа розробки з відкритим вихідним кодом для створення різних типів додатків, таких:

- Веб-додатки, веб-API і мікрослужби;
- Бессерверні функції в хмарі;
- Повністю хмарні додатки;
- Мобільні додатки;
- Класичні додатки: Windows WPF, Windows Forms, Універсальна платформа Windows (UWP);
- Ігри;
- Інтернет речі;
- Машинне навчання;
- Консольні додатки;
- Служби Windows [19].

До можливостей C # відноситься:

- Повна підтримка класів і ООП;
- Погоджений і чітко визначений набір базових типів;
- Підтримка автоматичної генерації XML-документації;
- Підтримка властивостей і подій в стилі Visual Basic;
- Можливість використання для написання динамічних web-сторінок ASP.NET і web-служб XML [21].

Програми C# виконуються в .NET, віртуальній системі виконання, що викликає загальномовне середовище виконання (CLR) і набір бібліотек класів. Середа CLR - це реалізація загальномовної інфраструктури мови (CLI), що є міжнародним стандартом, від корпорації Майкрософт. CLI є основою для створення середовищ виконання і розробки, в яких мови і бібліотеки прозоро працюють один з одним [18].

Вихідний код, написаний на мові C# компілюється в проміжний мова (IL), який відповідає специфікаціям CLI. Код на мові IL і ресурси, в тому числі растрові зображення і рядки, зберігаються в збірці, зазвичай з розширенням .dll. Збірка містить маніфест з інформацією про типи, версії, мови і регіональні параметри для цієї збірки.

Забезпечення взаємодії між мовами є ключовою особливістю .NET. Код IL, створений компілятором C#, відповідає специфікації загальних типів (CTS). Код IL, створений з коду на C #, може взаємодіяти з кодом, створеним з версій .NET для мов F#, Visual Basic, C++ і будь-яких інших мов, сумісних з CTS. Одна збірка може містити кілька модулів, написаних на різних мовах .NET, і всі типи можуть посилатися один на одного, як якщо б вони були написані на одній мові.

.NET також включає розширені бібліотеки. Ці бібліотеки підтримують безліч різних робочих навантажень. Вони впорядковані по просторах імен, які надають різні корисні можливості: від операцій файлового введення і виведення до управління рядками і синтаксичного аналізу XML, від платформ веб-додатків до елементів управління Windows Forms. Зазвичай додаток C# активно використовують бібліотеку класів .NET для вирішення типових задач.

Бібліотеки класів втілюють поняття загальної бібліотеки в .NET. Вони дозволяють поміщати корисні функції в модулі, які можуть використовуватися різними додатками. Вони також можуть використовуватися для підключення функцій, які не були потрібні або не були відомі під час запуску програми. Бібліотеки класів описуються в форматі файлу збірки .NET [20].

Існує три доступних для використання типу бібліотек класів:

1. Специфічні для платформи бібліотеки класів мають доступ до всіх API даної платформи (наприклад, .NET Framework, Xamarin iOS), але можуть використовуватися тільки додатками і бібліотеками, призначеними для цієї платформи.
2. Переносні бібліотеки класів мають доступ до підмножини API і можуть використовуватися додатками і бібліотеками, призначеними для декількох платформ.
3. Бібліотеки класів .NET Standard є об'єднанням специфічних для платформи і переносних бібліотек, і поєднують кращі характеристики обох типів.

2.2. Технології для розробки інтерфейсу користувача

Для створення графічних інтерфейсів за допомогою платформи .NET застосовуються різні технології - Window Forms, WPF, додатки для магазину Windows Store (для ОС Windows 8 / 8.1 / 10). Однак найбільш простий і зручною платформою досі залишається Window Forms [22].

Windows Forms - це технологія призначена для інтерфейсу користувача для .NET, що представляє собою набір керованих бібліотек, які спрощують виконання стандартних завдань, таких як читання з файлової системи і запис в неї. За допомогою середовища розробки, таких як Visual Studio, можна створювати інтелектуальні клієнтські програми Windows Forms, які відображають інформацію, запитують введення користувача і взаємодіють з віддаленими комп'ютерами по мережі [23].

У Windows Forms, форма - це візуальна поверхня, на якій виводиться інформація для користувача. Зазвичай додаток Windows Forms будується шляхом додавання елементів управління в форми і створення коду для реагування на дії користувача, такі як клацання миші або натискання клавіш. Елемент управління - це окремий елемент, призначений для користувача інтерфейсу та для відображення або введення даних.

При виконанні користувачем якої-небудь дії з формою або одним з її елементів управління створюється подія. Додаток реагує на ці події, як задано в коді, і обробляє події при їх виникненні.

У Windows Forms передбачено безліч елементів управління, які можна додавати в форми. Наприклад, елементи управління можуть відображати текстові поля, кнопки, списки, що розкриваються, перемикачі та навіть веб-сторінки. Якщо передбачені елементи управління не підходять для ваших цілей, в Windows Forms можна створювати власні призначені для користувача елементи управління за допомогою класу UserControl.

У Windows Forms є багатофункціональні елементи управління користувальницького інтерфейсу, що дозволяють емулювати функції таких складних додатків, як Microsoft Office. За допомогою елементів управління ToolStrip і MenuStrip ви можете створювати панелі інструментів і меню, які містять текст і зображення, відображають підменю і розміщують інші елементи управління, такі як текстові поля і поля зі списками.

Використовуючи функцію перетягування конструктора Windows Forms в Visual Studio, можна легко створювати додатки Windows Forms. Просто виділіть елемент управління за допомогою курсору і помістіть його на потрібне місце в формі. Для подолання труднощів, пов'язаних з вирівнюванням елементів управління, конструктор надає такі засоби, як лінії сітки і лінії прив'язки. За допомогою елементів управління FlowLayoutPanel, TableLayoutPanel і SplitContainer можна набагато швидше створювати складні макети форм.

Нарешті, якщо потрібно створити свої власні елементи призначеного для користувача інтерфейсу, простір імен System.Drawing містить широкий набір класів, необхідних для відтворення ліній, кіл та інших фігур безпосередньо на формі.

2.3. Середовище для розробки системи

Інтегроване середовище розробки або IDE - це додаток або програмне забезпечення, яке програмісти використовують для створення програмного продукту. IDE допомагає програмісту легко програмувати, надаючи всі комплексні засоби, необхідні для розробки програмного забезпечення. IDE може підвищити продуктивність програміста або розробника завдяки швидкому налаштуванню і різним інструментам. Без цього програмісту потрібно багато часу, щоб вибрати різні інструменти для вирішення своїх завдань [24].

В основному IDE складається з трьох частин: редактора вихідного коду, засоби автоматизації збирання (компілятора) і відладчика. Редактор вихідного коду - це те, де програмісти можуть писати код, тоді як інструмент автоматизації збирання використовується програмістами для компіляції коду, а відладчик використовується для тестування або налагодження програми для усунення будь-яких помилок в коді. Крім того, IDE також мають додаткові функції, такі як моделювання об'єктів і даних, модульне тестування, бібліотеку вихідного коду і багато іншого.

На сьогоднішній день найпопулярнішими IDE, які використовуються для розробки програмного забезпечення на C#, є Visual Studio, Project Rider, Eclipse aCute, Visual Studio Code і MonoDevelop [25].

Project Rider являє собою кроссплатформену .Net IDE. Вона підходить для використання під Windows, Linux, Mac OS X. Продукт базується на IntelliJ IDEA і Resharper.

У число переваг входить:

1. Підтримка C #, VB, XAML, HTML, JavaScript, TS і інших мов;
2. Дуже добре підходить для створення різного програмного забезпечення: ASP.Net, Xamarian та інше;

3. Потужна підтримка навігації і рефакторінга;
4. Прекрасно реалізована підтримка інтелектуальних поєднань клавіш;
5. Інтеграція з Visual Studio і Unity.

До недоліків відносять такі властивості:

1. Деяка частина функціоналу ще в процесі розробки, тому продукт містить в собі помилки і баги;
2. Висока вартість. Ціна за використання платформи - 139 USD в рік. При цьому є trial-версія і знижки для студентів.

Eclipse aCute - плагін для Eclipse IDE. Він полегшує розробку на C #. aCute дає можливість застосовувати редактор C #, до складу якого входить Eclipse IDE, що підтримує мови за допомогою сервера Omni-sharp.

У число переваг входить:

1. Плагін aCute виділяє синтаксис кольором;
2. Можливість розробляти повну версію основного проекту .Net, не виходячи з IDE;
3. Можна виконувати програми, розроблені із застосуванням MS test і xUnit.

До недоліків відносять такі властивості:

1. Висока складність освоєння для початківців;
2. Плагін розроблений невеликою спільнотою. Отже, немає ніяких гарантій, що він завжди буде працювати якісно.

Visual Studio Code - відмінний високопродуктивний легкий редактор, який пропонує хорошу підтримку завершення проекту. Працює на Node JS. Містить плагіни для VIM і Emacs.

У число переваг входить:

1. Працює на базі відкритого вихідного коду;
2. Відмінно працює на Mac, Linux, Windows;
3. Включає великий функціонал. У нього входить оглядач рішень, відладчик, область розширень і управління вихідним кодом;
4. Підтримка терміналу всередині вікна;

5. Чудово підходить для розробки на ядрі .Net.

У продукту є і певні недоліки:

1. Мінімалістичний, тому складний для сприйняття деякими розробниками;
2. Володіє невеликим функціоналом, тому не підходить для реалізації великих проектів.

MonoDevelop підходить для швидкої розробки настільних і web-додатків. IDE дозволяє переносити додатки .Net, написані в Visual Studio, на Linux і Mac OS X, оскільки підтримує єдину базу коду платформ.

До переваг платформи відносять наступне:

1. Мультиплатформеність;
2. Можливість налаштування продукту під кожного розробника;
3. Наявність відладчика і іншого корисного інструментарію;
4. Повністю підтримує популярну платформу для створення комп'ютерних ігор Unity 3D;
5. На 100% безкоштовний продукт.

До недоліків платформи відносять:

1. Невеликий функціонал;
2. Платформа не може підтримувати різні проекти.

Для розробки системи моніторингу та управління IP адресацією комп‘ютерної мережі в даній роботі було обрано Microsoft Visual Studio — серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ [26].

У число переваг Visual Studio входить наступне:

1. Середовище містить безліч інструментів, які дуже добре працюють на C#;

2. Наявність безкоштовної версії - Community Edition;
3. Найефективніше ПО для розробки на платформі .Net і C#;
4. Можливість зберігання даних в хмарі.

У продукту є і деякі недоліки:

1. Великі затрати ресурсів;
2. Складність при самостійному освоєнні [25].

2.4. Висновки до розділу

Для розробки системи моніторингу та управління IP адресацією комп‘ютерної мережі було використано декілька технологій, які виконують різні функції: технологія для розробки функціональної складової системи та технологія для розробки інтерфейсу користувача.

Для розробки функціональної складової системи було обрано об'єктно-орієнтовану мову програмування C#. C# дозволяє розробникам створювати безліч безпечних і надійних програм, які працюють на фреймворці .NET. Основними перевагами даного фреймворка є доступність, кросплатформеність, стабільність та широка бібліотека класів.

.NET - це фреймворк від Microsoft, який дозволяє використовувати одні і ті ж простори імен, бібліотеки і API для різних мов. .Net Framework - це набір вже скомпільованих бібліотек, звідки беруться методи і функції для запуску і розробки додатків.

Для розробки інтерфейсу користувача було обрано Windows Forms – технологію призначену для інтерфейсу користувача в фреймворці .NET, що представляє собою набір керованих бібліотек, які спрощують виконання стандартних завдань, таких як читання з файлової системи і запис в неї.

Безпосередньо для створення компонентів даної системи було обрано інтегроване середовище розробки Microsoft Visual Studio – середовище для написання, налагодження і складання коду, а також подальшої публікації додатків. Інтегроване середовище розробки (IDE) являє собою багатофункціональну

програму, яку можна використовувати для різних аспектів розробки програмного забезпечення. Крім стандартного редактора і відладчика, які існують в більшості середовищ IDE, Visual Studio включає в себе компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій для спрощення процесу розробки.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ МОНІТОРИНГУ КОМП'ЮТЕРНОЇ МЕРЕЖІ ТА УПРАВЛІННЯ IP АДРЕСАЦІЄЮ

Файлова структура проекту, яка зображена на рис. 3.1, містить декілька основних файлів, які відповідають за основну функціональність системи та її працездатність.

Рис. 3.1. Файлова структура системи

Файли, які складають основу системи:

- Form1.cs - це кодовий файл класу форми windows, в якому записуються необхідні методи, функції, а також методи, керовані подіями, і коди;
- Form1.Designer.cs - це файл конструктора, в якому ініціалізуються елементи форми. Якщо елемент перетягується в вікно форми, то цей елемент буде автоматично проініціалізован в цьому класі;
- Program.cs - це файл коду за замовчуванням, який створюється при створенні нової програми в Visual Studio. Цей код буде містити код запуску для додатка в цілому.

Етапи розробки системи моніторингу та управління IP адресацією комп’ютерної мережі показано на рис. 3.2.

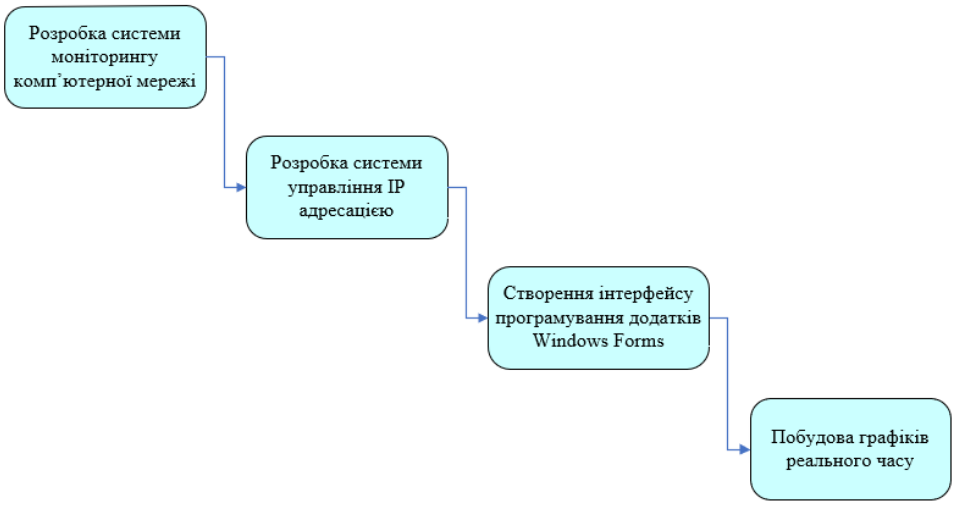

Рис. 3.2. Етапи розробки системи

3.1. Розробка функціональної складової системи

Функціональну складову системи моніторингу та управління IP адресацією комп’ютерної мережі можна розділити на 2 підсистеми:

- система моніторингу комп’ютерної мережі;
- система управління IP адресацією комп’ютерної мережі.

Кожна підсистема відповідає за свій функціонал, має різні методи, працює з різними даними, і в той же час об’єднані в одну систему. Функціональна складова системи створена в файлі Form1.cs.

3.1.1. Розробка системи моніторингу комп’ютерної мережі

При запуску програми система автоматично виконує методи, які відповідають за моніторинг мережі. Для цього система спочатку виявляє всі мережеві адаптери комп’ютера, і для кожного адаптера викликає методи для визначення пропускної здатності, затримки передачі та частоти втрати пакетів адаптера.

Всі дані, які відображає система, оновлюються кожну секунду. Це створено за допомогою змінної Updator типу Timer, що створює подію через заданий період з можливістю створення повторюваних подій. Updator має поле Interval, що задає інтервал в мілісекундах, після закінчення якого виникає повторення роботи функції wirelssUpdator_Tick, яка знаходиться в полі Tick (рис. 3.3). Функція wirelssUpdator_Tick викликає в свою чергу функцію BandwidthCalculator, що є головною в функціональній складовій системи моніторингу комп'ютерної мережі.

```
this.Updator.Interval = 1000;
this.Updator.Tick += new System.EventHandler(this.wirelssUpdator_Tick);
```

Рис. 3.3. Програмна реалізація оновлення даних кожної секунди

3.1.1.1. Створення методів для виявлення мережевих адаптерів комп'ютера

Система має функцію Form1_Load для виявлення мережевих адаптерів комп'ютера. Інтерфейс IEnumerable надає нумератор, який підтримує простий перебір елементів. За допомогою цього нумератора система перебирає елементи типу NetworkInterface. Елементи, які мають тип мережевого інтерфейсу Ethernet або Wireless80211 (інтерфейс, що використовує бездротове підключення), додаються до інтерфейсу IEnumerable колекції NetworkInterface за допомогою методу GetAllNetworkInterfaces(). Ці елементи і є мережевими адаптерами комп'ютера.

Після виявлення адаптерів, система додає ці елементи з інтерфейсу IEnumerable до розгортуючого списку за допомогою циклу foreach. Процес програмної реалізації виявлення мережевих адаптерів зображено на рис. 3.4.

```
private void Form1_Load(object sender, EventArgs e)
{
    IEnumerable<NetworkInterface> nics = NetworkInterface.GetAllNetworkInterfaces().Where(network =>
    (network.NetworkInterfaceType == NetworkInterfaceType.Ethernet ||
    network.NetworkInterfaceType == NetworkInterfaceType.Wireless80211));

    cmbAdptors.DisplayMember = "Description";
    cmbAdptors.ValueMember = "Id";
    foreach (NetworkInterface item in nics)
    {
        cmbAdptors.Items.Add(item);
    }

    if (cmbAdptors.Items.Count > 0)
        cmbAdptors.SelectedIndex = 0;
}
```

Рис. 3.4. Програмна реалізація виявлення мережевих адаптерів

Результат роботи методів для виявлення мережевих адаптерів комп'ютера показано на рис. 3.5.

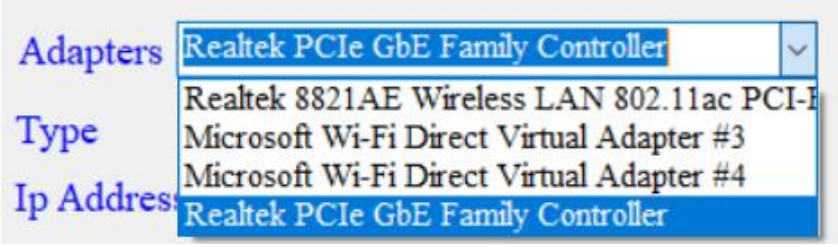

Рис. 3.5. Результат роботи методів для виявлення мережевих адаптерів

Для роботи з певним адаптером створена змінна (рис. 3.6.) типу NetworkInterface, яка зберігає значення вибраного адаптера.

```
NetworkInterface slectedNic = cmbAdptors.SelectedItem as NetworkInterface;
```

Рис. 3.6. Програмна реалізація змінної, яка зберігає значення вибраного адаптера

3.1.1.2. Створення методів для визначення пропускної здатності адаптера

IP адреса адаптера визначається за допомогою об'єкта типу UnicastIPAddressInformationCollection, який містить колекцію відомостей про адреси

мережевих інтерфейсів. В дану колекцію вносяться інформація, що описує конфігурацію мережевого інтерфейсу певного адаптера, за допомогою методу GetIPProperties(), як показано на рис. 3.7.

```
UnicastIPAddressInformationCollection ipInfo = slectedNic.GetIPProperties()
```

Рис. 3.7. Програмна реалізація визначення IP адреси адаптера

Для визначення MAC-адреси (фізичної адреси) даного адаптера, типу та операційного стану мережевого інтерфейсу використані стандартні методи GetPhysicalAddress, NetworkInterfaceType та OperationalStatus відповідно. Програмна реалізація показана на рис. 3.8.

```
lblType.Text = selecNic.NetworkInterfaceType.ToString();
lblOp.Text = selecNic.OperationalStatus.ToString();

PhysicalAddress macData = selecNic.GetPhysicalAddress();
lblMAC.Text = macData.ToString();
```

Рис. 3.8. Програмна реалізація визначення MAC-адреси адаптера, типу та операційного стану адаптера

Визначення часу, що залишився в секундах, протягом якого дійсна IP адреса певного адаптера відбувається за допомогою методу AddressValidLifetime, як показано на рис. 3.9. Час в секундах зберігається в об’єкті типу TimeSpan, який представляє різницю між поточним часом та часом, що залишився.

```
TimeSpan addressLifeTime = TimeSpan.FromSeconds(ipInfo.AddressValidLifetime);
```

Рис. 3.9. Програмна реалізація визначення часу протягом якого дійсна IP адреса

Метод GetIPv4Statistics використовується для визначення відомостей про мережевий трафік для інтерфейсу адаптера. Змінна interfaceData об’єкту типу IPv4InterfaceStatistics зберігає дані цього методу.

Для визначення числа отриманих байтів адаптером використано метод interfaceData. Щоб визначити швидкість передачі даних, знайдена різниця між кількістю отриманих даних на теперішній час та секундою раніше. Дана різниця розділена на 1024, щоб швидкість вимірювалась в KB/s (килобайти на секунду). Програмна реалізація отриманих байтів та швидкості передачі даних адаптером показано на рис. 3.10.

```
IPv4InterfaceStatistics interfaceData = selecNic.GetIPv4Statistics();
int bytesRecivedSpeedValue = (int)(interfaceData.BytesReceived - double.Parse(lblReceived.Text)) / 1024;
lblReceived.Text = interfaceData.BytesReceived.ToString();
lblSpeed.Text = bytesRecivedSpeedValue.ToString() + " KB/s";
```

Рис. 3.10. Програмна реалізація визначення отриманих байтів та швидкості передачі даних

Результат роботи методів для визначення пропускної здатності адаптера показано на рис. 3.11.

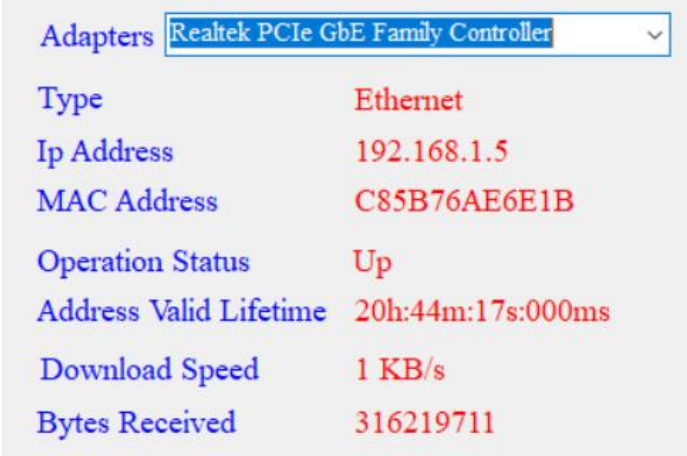

Рис. 3.11. Результат роботи методів для визначення пропускної здатності

3.1.1.3. Створення методів для визначення затримки передачі та частоти втрати пакетів

Об’єкт класу Ping використовується, щоб визначити, чи може служба успішно зв'язатися з віддаленим вузлом. Для визначення затримки передачі даних створена змінна ping_sender класу Ping та викликан метод Send, який робить спробу відправки повідомлення запиту перевірки зв'язку ICMP на віддалену IP адресу (дана система використовує IP адресу google.com – 8.8.8.8) і отримання від нього відповідного повідомлення відповіді перевірки зв'язку ICMP.

ICMP — мережевий протокол, що входить в стек протоколів TCP/IP. В основному ICMP використовується для передачі повідомлень про помилки й інші виняткові ситуації, що виникли при передачі даних. Також на ICMP покладаються деякі сервісні функції, зокрема на основі цього протоколу заснована дія таких загальновідомих утиліт як ping та traceroute [27].

Відомості про стан і дані, отримані в результаті операції Send записується в змінну ping_reply типу PingReply. Для ping_reply викликано метод RoundtripTime, що повертає час у мілісекундах, витрачений на відправку запиту перевірки зв'язку по протоколу ICMP і отримання відповідного повідомлення з відповіддю перевірки зв'язку по протоколу ICMP. Програмна реалізація для визначення затримки передачі показано на рис. 3.12.

```
string ping_adress = "8.8.8.8";
Ping ping_sender = new Ping();
PingReply ping_reply = ping_sender.Send(ping_adress);
lblPing.Text = ping_reply.RoundtripTime.ToString() + " ms";
```

Рис. 3.12. Програмна реалізація визначення затримки передачі даних

Результат роботи методу для визначення затримки передачі даних показано на рис. 3.13.

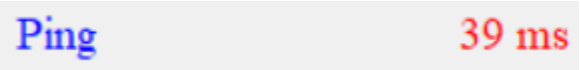

Рис. 3.13. Результат роботи методу для визначення затримки передачі

Для визначення частоти втрати пакетів створена функція під назвою get_loss, яка повертає кількість втрачених пакетів у відсотках. До вхідних параметрів функції входить IP адреса хоста на який буде відправлятися пакети та кількість пакетів. В даній системі пакети відправляються на локальний хост з адресою 127.0.0.1 в кількості 15 пакетів.

Створено змінна options класу PingOptions, яка використовується для управління передачею пакетів даних Ping, та використано до неї метод DontFragment, що забороняє відправку даних декількома пакетами. Система також використовує метод Send, але крім IP адресу, також передано буфер даних, що складається з 32 байт, тайм-аут між відправкою пакетів (60 мс) та змінна options. Кількість пакетів, які не дійшли до хоста, рахує змінна failed. Щоб визначити кількість втрачених пакетів, система виконує ділення змінної failed на загальну кількість відправлених пакетів. Дане значення домножене на 100, щоб результат був у відсотках. Програмна реалізація методу показана на рис. 3.14, а результат виконання на рис. 3.15.

```
private double get_loss(String host, int pingAmount)
{
    Ping pingSender = new Ping();
    PingOptions options = new PingOptions();

    // Данные нельзя отправлять несколькими пакетами
    options.DontFragment = true;

    // Создание буфера из 32 байтов данных для передачи.
    string data = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
    byte[] buffer = Encoding.ASCII.GetBytes(data);

    int timeout = 60;
    int failed = 0;

    // Зациклить количество раз, чтобы пинговать
    for (int i = 0; i < pingAmount; i++)
    {
        PingReply reply = pingSender.Send(host, timeout, buffer, options);
        if (reply.Status != IPStatus.Success)
        {
            failed += 1;
        }
    }

    // Вернуть процент
    double percent = (failed / pingAmount) * 100;
    return percent;
}
```

Рис. 3.14. Програмна реалізація визначення частоти втрати пакетів

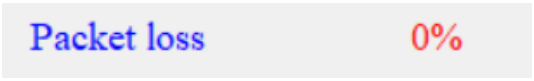

Рис. 3.15. Результат роботи методу для визначення частоти втрати пакетів

3.1.2. Розробка системи управління IP адресацією

Система управління IP адресацією працює таким чином, що користувач вводить IP адресу та маску підмережі та отримує інформацію про адресу мережі, широкомовну адресу, зворотню маску, кількість та діапазон хостів в мережі, та зайняті IP адреси в даній мережі.

IP адреса та маска підмережі вводяться в відповідні поля, які зображені на рис. 3.16. Маску підмережі дозволено вводити в повній та скороченій формах, наприклад 255.255.224.0 або 19.

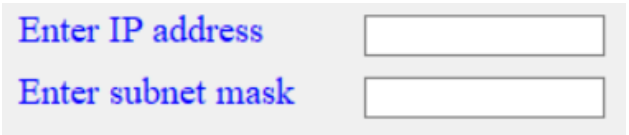

Рис. 3.16. Поля для вводу IP адреси та маски підмережі

Система має кнопку, яка перевіряє коректність вводу IP адреси та маски підмережі, і в разі їх правильності починає розрахунок. Якщо дані введені не вірно, то система попереджує користувача та очікує поки IP адреса та маска підмережі будуть введені коректно.

Для визначення адреси мережі та широкомовної адреси створена функція, вхідними параметрами якої є IP адреса та маска підмережі. Система визначає, яка частина IP адреси відноситься до мережі, а яка до хосту. Для знаходження адреси мережі система підставляє 0 в ту частину IP адреси, яка відноситься до хосту, а для знаходження широкомовної адреси підставляє 1. Також система віднімає маску підмережі від 255.255.255.255 і в результаті знаходить зворотню маску.

Діапазон хостів в мережі система визначає таких чином: початкова адреса в діапазоні визначається додаванням 1 до останнього біта адреси мережі, а кінцева адреса – відніманням 1 від останнього біта широкомовної адреси. Визначивши діапазон хостів система розраховує кількість хостів в мережі, як показано на рис. 3.17.

```
public static int Get_Hosts(IPAddress net_address, IPAddress broadcast_address)
{
    byte[] NetBytes = net_address.GetAddressBytes();
    byte[] BroadcastBytes = broadcast_address.GetAddressBytes();

    int amount = 1;
    int temp = 0;
    for (int i = 0; i < NetBytes.Length; i++)
    {
        if (NetBytes[i] == BroadcastBytes[i])
        {
            amount = 1;
        }
        else if (NetBytes[i] == 0)
        {
            temp = (byte)(BroadcastBytes[i] - (NetBytes[i])) + 1;
            amount *= temp;
        }
        else
        {
            temp = (byte)(BroadcastBytes[i] - (NetBytes[i])) + 3;
            amount = (amount * temp) - 2;

        }
        temp = 0;
    }
    return amount;
}
```

Рис. 3.17. Програмна реалізація визначення кількісті хостів

Зайняті IP адреси в даній мережі система знаходить за допомогою команди arp з аргументом -a в командному рядку комп’ютера. Команда arp уможливлює

створення, редагування і відображення зіставлень фізичних адрес відомим IPv4-адрес. Команда arp -a перераховує всі пристрої, які в даний момент представлені в ARP-кеші вузла, а також IPv4-адреси. Результат команди arp з аргументом -a система записує в строкову змінну, як показано на рис. 3.18.

```
ProcessStartInfo arp_a = new ProcessStartInfo("arp", " -a");
arp_a.UseShellExecute = false;
arp_a.RedirectStandardOutput = true;
arp_a.CreateNoWindow = true;
var proc = Process.Start(arp_a);

string s = proc.StandardOutput.ReadToEnd();
```

Рис. 3.18. Програмна реалізація визначення зайнятих хостів

ARP — комунікаційний протокол, призначений для перетворення IP адрес (адрес мережевого рівня) в MAC адреси (адреси канального рівня) в мережах TCP/IP. Перетворення адрес виконується шляхом пошуку за таблицею. Ця таблиця називається ARP-таблицею, зберігається у пам'яті й містить рядки для кожного вузла мережі. В двох стовпчиках містяться IP та Ethernet адреси [28].

В змінній з результат записано IP адреси, їх тип та фізичні адреси, тому система перетворює дану змінну (рис. 3.19.) таким чином, щоб в результаті було записано тільки зайняті IP адреси.

```
RegexOptions options = RegexOptions.None;
Regex regex = new Regex("[ ]{2,}", options);
s = regex.Replace(s, " ");

string[] subs = s.Split(' ');
string usedIp = "";

for (int i = 10; i < subs.Length; i += 3)
{
    usedIp = usedIp + subs[i] + "\n";
}

lblUsed.Text = usedIp;
```

Рис. 3.19. Програмна реалізація перетворення рядка з результатом

Результат роботи системи управління IP адресацією показано на рис. 3.20.

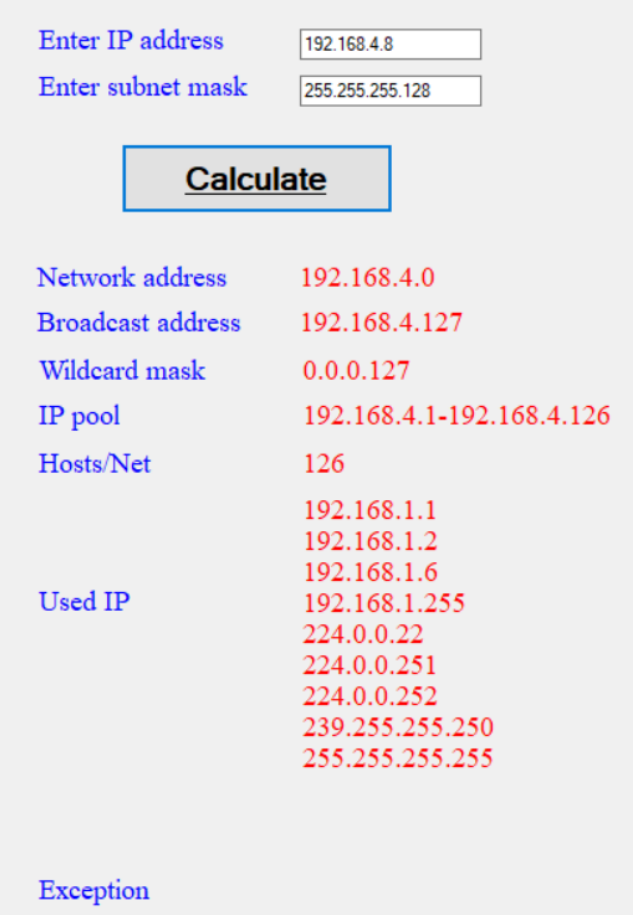

Рис. 3.20. Результат роботи системи управління IP адресацією

3.2. Розробка інтерфейсу користувача

Файл Form1.Designer.cs відповідає за весь інтерфейс користувача в даній системі. Цей файл містить відповідні вбудовані бібліотеки і класи для створення інтерфейсу користувача. Розробку інтерфейсу можна розділити на 2 частини:

- Створення інтерфейсу програмування додатків Windows Forms;
- Побудова графіків реального часу – цей процес є складнішим, оскільки немає вбудованих бібліотек для створення графіків в реальному часі.

3.2.1. Створення інтерфейсу програмування додатків Windows Forms

Більшість інформації та результатів, які бачить користувач знаходяться в полі типу Label. Майже всі поля Label створені по парах (рис. 3.21): одне поле містить опис інформації або даних, які розраховує система, та друге поле, яке розташоване праворуч – відображає результат розрахунків.

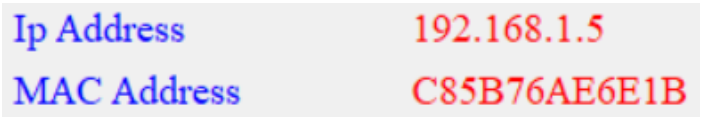

Рис. 3.21. Приклад пари полів label

Поля Label проініціалізові та описані в файлі Form1.Designer.cs, як це показано на рис. 3.22, але всі властивості можливо змінити в інших файлах системи. Поля, що відображають результат розрахунків, змінюють властивість Text (рис. 3.23), яка описує вміст Label, в файлі Form1.cs.

```
this.label4.AutoSize = true;
this.label4.BackColor = System.Drawing.Color.Transparent;
this.label4.Font = new System.Drawing.Font("Times New Roman", 13F);
this.label4.ForeColor = System.Drawing.Color.Blue;
this.label4.Location = new System.Drawing.Point(29, 171);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(140, 25);
this.label4.TabIndex = 14;
this.label4.Text = "MAC Address";
//
// lblMAC
//
this.lblMAC.AutoSize = true;
this.lblMAC.BackColor = System.Drawing.Color.Transparent;
this.lblMAC.Font = new System.Drawing.Font("Times New Roman", 13F);
this.lblMAC.ForeColor = System.Drawing.Color.Red;
this.lblMAC.Location = new System.Drawing.Point(268, 171);
this.lblMAC.Name = "lblMAC";
this.lblMAC.Size = new System.Drawing.Size(50, 25);
this.lblMAC.TabIndex = 15;
this.lblMAC.Text = "N/A";
```

Рис. 3.22. Описання полів Label

```
lblMAC.Text = macData.ToString();
```

Рис. 3.23. Зміна властивості Text в файлі Form1.cs

Система також містить розгортуючий список типу ComboBox, що дозволяє користувачу вибирати адаптер, інформацію якого він хоче отримати. Ініціалізація та описання списку (рис. 3.24) відбувається в Form1.Designer.cs, а заповнення – аналогічно з Label у файлі Form1.cs в функції Form1_Load.

```
this.cmbAdptors.BackColor = System.Drawing.Color.White;
this.cmbAdptors.Font = new System.Drawing.Font("Times New Roman", 11F);
this.cmbAdptors.ForeColor = System.Drawing.Color.Black;
this.cmbAdptors.FormattingEnabled = true;
this.cmbAdptors.Location = new System.Drawing.Point(130, 53);
this.cmbAdptors.Name = "cmbAdptors";
this.cmbAdptors.Size = new System.Drawing.Size(379, 28);
this.cmbAdptors.TabIndex = 1;
this.cmbAdptors.SelectedIndexChanged += new System.EventHandler(this.cmbAdptors_SelectedIndexChanged);
```

Рис. 3.24. Описання розгортаючого списку

Система управління IP адресацією має кнопку типу Button, яка перевіряє коректність вводу даних та починає розрахунок. Описання кнопки в Form1.Designer.cs показано на рис. 3.25. Кнопка має властивість Click, що виконує певну дію при натисканні на неї. В даній системі властивість Click викликає функцію button_calculate_Click, яка починає розрахунок даних і описана в файлі Form1.cs.

```
this.button_calculate.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F, ((S
this.button_calculate.Location = new System.Drawing.Point(1415, 196);
this.button_calculate.Margin = new System.Windows.Forms.Padding(4);
this.button_calculate.Name = "button_calculate";
this.button_calculate.Size = new System.Drawing.Size(233, 55);
this.button_calculate.TabIndex = 34;
this.button_calculate.Text = "Calculate";
this.button_calculate.UseVisualStyleBackColor = true;
this.button_calculate.Click += new System.EventHandler(this.button_calculate_Click);
```

Рис. 3.25. Описання кнопки для розрахунку системи

3.2.2. Побудова графіків реального часу

Затримка та швидкість передачі даних в системі відображаються не тільки в числовому виді, а і у вигляді графіків реального часу. Для побудови графіків використано клас Chart, що має всі властивості, методи і події керування

діаграмами, та 2 допоміжні класи типу ChartArea, що представляє область діаграми, і Series - точка даних на графіку.

Створено масив на 60 елементів, так як графік реального часу відображає інформацію за останні 60 секунд, та 2 потоки, щоб графіки будувались одночасно. Запис даних до масиву відбувається з кінця, так як і побудова графіків. Після того як дані записано до масиву відбувається дія методу Copy, що копіює діапазон елементів з масиву Array, починаючи з заданого індексу джерела, і вставляє його в інший масив Array, починаючи з заданого індексу призначення. Тобто метод Copy копіює всі елементи масиву крім першого та вставляє їх в цей же масив, починаючи вже з першого. Після цього виконується функція UpdatePingChart або UpdateSpeedChart. Програмна реалізація заповнення масива показана на рис. 3.26.

```
pingArray[pingArray.Length - 1] = Ping_chart_val;

Array.Copy(pingArray, 1, pingArray, 0, pingArray.Length - 1);

if (pingChart.IsHandleCreated)
{
    this.Invoke((MethodInvoker)delegate { UpdatePingChart(); });
}
```

Рис. 3.26. Програмна реалізація заповнення масива

Функції UpdatePingChart та UpdateSpeedChart використовуються для позначення точок на графіку за допомогою класу Series. Точки по координаті Y ставляться за значенням елемента масиву, як показано на рис. 3.27, а по X – за відповідним номером елемента в масиві.

```
private void UpdatePingChart()
{
    pingChart.Series["Series1"].Points.Clear();

    for (int i = 0; i < pingArray.Length - 1; ++i)
    {
        pingChart.Series["Series1"].Points.AddY(pingArray[i]);
    }
}
```

Рис. 3.27. Програмна реалізація функції UpdatePingChart

Результат розробки графіків реального часу показано на рис. 3.28.

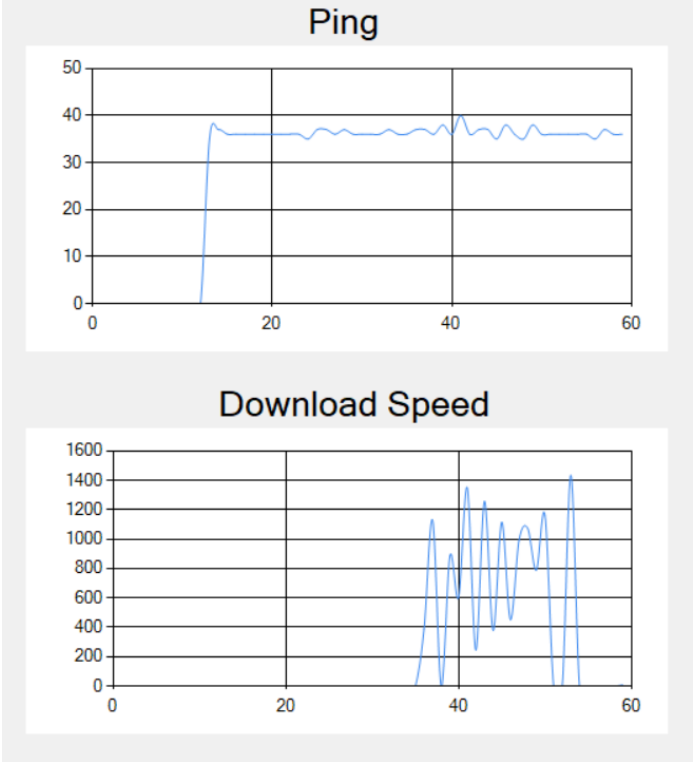

Рис. 3.28. Графіки реального часу

Результат роботи програмної системи моніторингу та управління IP адресацією комп‘ютерної мережі показано на рис. 3.29.

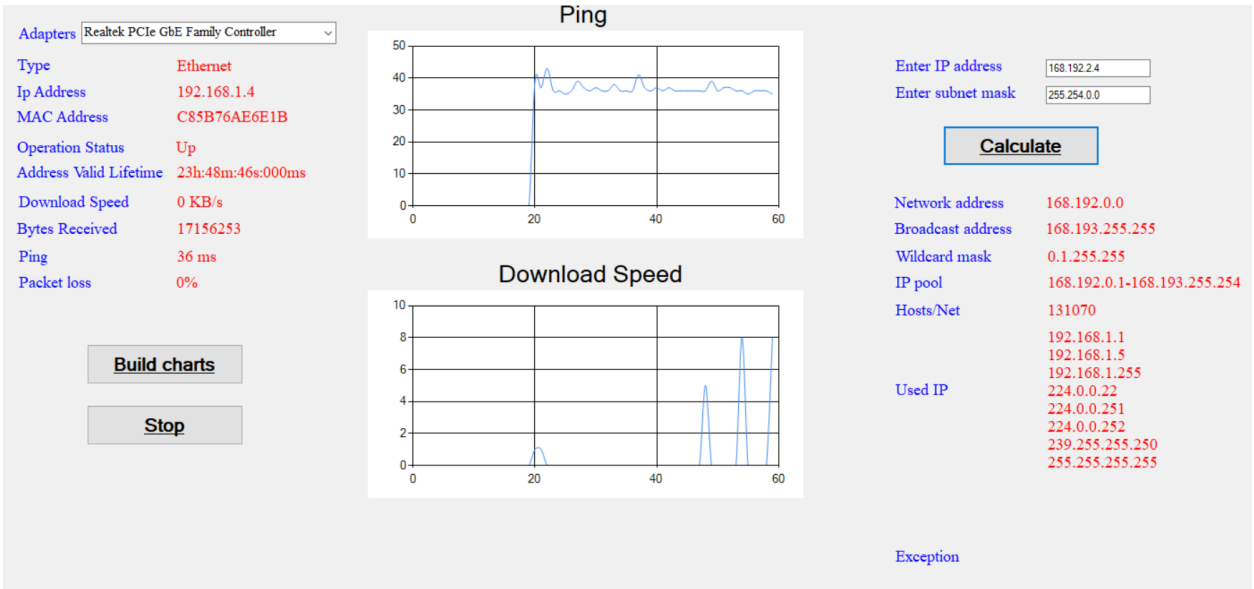

Рис. 3.29. Результат роботи системи

3.3. Висновки до розділу

Основними етапами розробки системи моніторингу та управління IP адресацією комп‘ютерної мережі є створення функціональної складової системи моніторингу комп’ютерної мережі та системи управління IP адресацією, розробка інтерфейсу користувача.

Для розробки функціональної складової системи моніторингу комп’ютерної мережі були створенні методи:

- для виявлення мережевих адаптерів комп’ютера;
- для визначення пропускної здатності адаптера;
- для визначення затримки передачі та частоти втрати пакетів.

Для розробки функціональної складової системи управління IP адресацією були створенні методи:

- для вводу IP адреси та маски підмережі;
- для визначення адреси мережі та широкомовної адреси;
- для визначення зворотньої маски;
- для визначення кількості та діапазону хостів в мережі;
- для визначення зайнятих IP адрес в даній мережі.

Графічний інтерфейс користувача створено за допомогою технології Windows Forms. Інтерфейс користувача має поля для вводу та виводу інформації, кнопку для розрахунку даних, кнопки для побудови та зупинення графіків, і 2 графіки реального часу.

В результаті створена система моніторингу та управління IP адресацією комп’ютерної мережі, яку можуть використовувати будь-які користувачі комп’ютера для визначення проблем з мережею та управління IP адресацією, і взаємодіяти з системою за допомогою створеного інтерфейсу користувача.

ВИСНОВКИ

Комп'ютерна мережа є сьогодні практично в кожній фірмі та компанії. Мережа вже не вважається чимось розкішним, а є чудовою нагодою ефективно оптимізувати роботу, виробництво, об'єднавши всі комп'ютери в єдину систему.

Постійний контроль за роботою локальної мережі, необхідний для підтримки її в працездатному стані. Контроль - це необхідний етап, який повинен виконуватися при управлінні мережею.

При розглядані існуючих систем моніторингу та управління IP адресацією комп‘ютерної мережі було виявлено деякі недоліки, як надмірна функціональність, необхідність стабільної оплати за використання сервісів та додатків, та складність використання. Отже, виникла необхідність створення власної системи для уникнення зазначених проблем.

В процесі розробки системи були використані такі технології:

- C# - сучасна об'єктно- і компонентно-орієнтована мова програмування. C# дозволяє розробникам створювати безліч безпечних і надійних програм, які працюють на фреймворці .NET;
- Windows Forms - це технологія призначена для інтерфейсу користувача для .NET, що представляє собою набір керованих бібліотек, які спрощують виконання стандартних завдань, таких як читання з файлової системи і запис в неї;
- Microsoft Visual Studio – інтегроване середовище, яке було обране для створення компонентів даної системи і використовується для написання, налагодження і складання коду, а також подальшої публікації додатків.

Основні можливості, які надає система моніторингу та управління IP адресацією комп‘ютерної мережі:

- ідентифікація мережевих адаптерів доступних на комп'ютері;
- аналіз пропускну здатність обраного адаптера;

- визначення величини затримки передачі та кількості втрачених пакетів;
- візуалізація зібраних даних в графіки реального часу для подальшого аналізу;
- визначення адреси мережі та широкомовної адреси;
- розрахунок кількість хостів і діапазон допустимих IP адрес в мережі;
- відображення інформації щодо зайнятих IP адрес в даній мережі.

Для використання системи моніторингу та управління IP адресацією комп‘ютерної мережі її потрібно завантажити на пристрій користувача та запустити. Додаткового налаштування та завантаження система не потребує.

Даний функціонал системи в майбутньому можна розширити додавши:

- веб-інтерфейс, що не потребує завантаження системи на пристрій користувача;
- збереження інформації у файлі на пристрої або хмарі.

Перевагами розробленої системи є безкоштовне використання, легкість в розумінні та невеликий функціонал, що показує основну інформацію про мережу. Її використання підвищить ефективність роботи користувачам, які не мають знань для роботи з системами з великим функціоналом.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Что такое мониторинг сети? [Електронний ресурс]. Режим доступу: https://www.cisco.com/ (дата звернення 22.05.2021) – Назва з екрану.
2. Мониторинг сети [Електронний ресурс]. Режим доступу: https://www.skachatreferat.ru/ (дата звернення 22.05.2021) – Назва з екрану.
3. IP address management [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/ (дата звернення 22.05.2021) – Назва з екрану.
4. Компьютерные сети [Електронний ресурс]. Режим доступу: https://works.doklad.ru/ (дата звернення 22.05.2021) – Назва з екрану.
5. Администрирование информационных систем [Електронний ресурс]. Режим доступу: http://inftis.narod.ru/ (дата звернення 22.05.2021) – Назва з екрану.
6. Пять ключевых функций систем мониторинга производительности сети [Електронний ресурс]. Режим доступу: https://networkguru.ru/ (дата звернення 22.05.2021) – Назва з екрану.
7. Методы мониторинга и обеспечения безопасности для поддержания работоспособности [Електронний ресурс]. Режим доступу: http://www.corem.ru/ (дата звернення 22.05.2021) – Назва з екрану.
8. Обзор методов анализа и мониторинга сетевого трафика [Електронний ресурс]. Режим доступу: http://it-bloknot.ru/ (дата звернення 22.05.2021) – Назва з екрану.
9. Сравнительный анализ популярных систем мониторинга сетевого оборудования [Електронний ресурс]. Режим доступу: http://proceedings.spiiras.nw.ru/ (дата звернення 22.05.2021) – Назва з екрану.
10. 7 бесплатных программ для мониторинга сети и серверов [Електронний ресурс]. Режим доступу: https://networkguru.ru/ (дата звернення 22.05.2021) – Назва з екрану.

11. Сравнение систем мониторинга Zabbix vs Nagios [Електронний ресурс]. Режим доступу: http://www.netping.ru/ (дата звернення 22.05.2021) – Назва з екрану.

12. Zabbix vs Nagios vs Cacti [Електронний ресурс]. Режим доступу: https://www.psychz.net/ (дата звернення 22.05.2021) – Назва з екрану.

13. Руководство по управлению системой: Сети и средства связи [Електронний ресурс]. Режим доступу: http://www.regatta.cs.msu.su/ (дата звернення 22.05.2021) – Назва з екрану.

14. Операционные системы и среды [Електронний ресурс]. Режим доступу: https://kursksu.ru/sveden/ (дата звернення 22.05.2021) – Назва з екрану.

15. Сети, подсети, классы подсетей. Таблица подсетей. [Електронний ресурс]. Режим доступу: https://rtfm.co.ua/ (дата звернення 22.05.2021) – Назва з екрану.

16. Обратная маска [Електронний ресурс]. Режим доступу: https://habr.com/ru/ (дата звернення 22.05.2021) – Назва з екрану.

17. Общие архитектуры веб-приложений [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/ (дата звернення 22.05.2021) – Назва з екрану.

18. Краткий обзор языка C# [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/ (дата звернення 22.05.2021) – Назва з екрану.

19. Введение в .NET [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/ (дата звернення 22.05.2021) – Назва з екрану.

20. Библиотеки классов .NET [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/ (дата звернення 22.05.2021) – Назва з екрану.

21. Где в современном мире применяется C# [Електронний ресурс]. Режим доступу: https://levelup.ua/ (дата звернення 22.05.2021) – Назва з екрану.

22. Введение в Windows Forms [Електронний ресурс]. Режим доступу: https://metanit.com/ (дата звернення 22.05.2021) – Назва з екрану.

23. Руководство по классическим приложениям (Windows Forms .NET) [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/ (дата звернення 22.05.2021) – Назва з екрану.

24. Top IDEs for C or C++ Developers in 2021 & Beyond! [Електронний ресурс]. Режим доступу: https://blog.eduonix.com/ (дата звернення 22.05.2021) – Назва з екрану.

25. Лучшие IDE для C#-разработчика [Електронний ресурс]. Режим доступу: https://spb.deveducation.com/ (дата звернення 22.05.2021) – Назва з екрану.

26. Microsoft Visual Studio [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/ (дата звернення 22.05.2021) – Назва з екрану.

27. ICMP [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/ дата звернення 22.05.2021) – Назва з екрану.

28. ARP [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/ (дата звернення 22.05.2021) – Назва з екрану.

ДОДАТКИ

Додаток А

Програмна реалізація файлу Form1.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Net;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.NetworkInformation;
using System.Threading;
using System.Net.Sockets;
using System.Diagnostics;
using System.Text.RegularExpressions;

namespace NetworkMonitoring
{
   public partial class Form1 : Form
   {
      // построение графика
      bool Build = true;

      /////////// для Packetloss
      long Ping_chart_val = 0;
      long Speed_chart_val = 0;

      private Thread pingThread;
      private double[] pingArray = new double[60];
      private Thread speedThread;
      private double[] speedArray = new double[60];
      ////////////////////


      NetworkInterface slectedNic;
      UnicastIPAddressInformation uniCastIPInfo;


      public Form1()
      {
         InitializeComponent();
      }

      //Функция определения Packet loss
      private double get_loss(String host, int pingAmount)
      {
         Ping pingSender = new Ping();
         PingOptions options = new PingOptions();

         // Данные нельзя отправлять несколькими пакетами
         options.DontFragment = true;

         // Создание буфера из 32 байтов данных для передачи.
         string data = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
         byte[] buffer = Encoding.ASCII.GetBytes(data);

         int timeout = 60;
         int failed = 0;
```

```
        // Зациклить количество раз, чтобы пинговать
    for (int i = 0; i < pingAmount; i++)
            {
        PingReply reply = pingSender.Send(host, timeout, buffer, options);
        if (reply.Status != IPStatus.Success)
        {
            failed += 1;
        }
    }

    // Вернуть процент
    double percent = (failed / pingAmount) * 100;
    return percent;
}

//функ для определения Network и Broadcast адресов
public static IPAddress Get_Network_Broadcast_Address(IPAddress address, IPAddress subnetMask, bool Net_address)
{
    byte[] ipAdressBytes = address.GetAddressBytes();
    byte[] subnetMaskBytes = subnetMask.GetAddressBytes();

    if (Net_address)
    {
        byte[] networkAddress = new byte[ipAdressBytes.Length];
        for (int i = 0; i < networkAddress.Length; i++)
        {
            networkAddress[i] = (byte)(ipAdressBytes[i] & (subnetMaskBytes[i]));
        }
        return new IPAddress(networkAddress);
    }
    else
    {
        byte[] broadcastAddress = new byte[ipAdressBytes.Length];
        for (int i = 0; i < broadcastAddress.Length; i++)
        {
            broadcastAddress[i] = (byte)(ipAdressBytes[i] | (subnetMaskBytes[i] ^ 255));
        }
        return new IPAddress(broadcastAddress);
    }
}


//функ для преобразования с короткой в полную маску
public static string Get_Mask(string netMask)
{
    string subNetMask = string.Empty;
    int calSubNet = 0;
    double result = 0;
    if (!string.IsNullOrEmpty(netMask))
    {
        calSubNet = 32 - Convert.ToInt32(netMask);
        if (calSubNet >= 0 && calSubNet <= 8)
        {
            int firOctet = 8 - (8 - calSubNet);
            for (int ipower = 7; ipower >= firOctet; ipower--)
            {
                result += Math.Pow(2, ipower);
            }
            subNetMask = "255.255.255." + Convert.ToString(result);
        }
        else if (calSubNet > 8 && calSubNet <= 16)
        {
```

```
                    int secOctet = 8 - (16 - calSubNet);
                for (int ipower = 7; ipower >= secOctet; ipower--)
                {
                    result += Math.Pow(2, ipower);
                }

                subNetMask = "255.255." + Convert.ToString(result) + ".0";
            }
            else if (calSubNet > 16 && calSubNet <= 24)
            {
                        int thirdOctet = 8 - (24 - calSubNet);
                for (int ipower = 7; ipower >= thirdOctet; ipower--)
                {
                    result += Math.Pow(2, ipower);
                }
                subNetMask = "255." + Convert.ToString(result) + ".0.0";
            }
            else if (calSubNet > 24 && calSubNet <= 32)
            {
                int fourthOctet = 8 - (32 - calSubNet);
                for (int ipower = 7; ipower >= fourthOctet; ipower--)
                {
                    result += Math.Pow(2, ipower);
                }
                subNetMask = Convert.ToString(result) + ".0.0.0";
            }
        }

        return subNetMask;
    }

//функ для определения обратной маски
    public static IPAddress Get_Wilcard(IPAddress NetAdress, IPAddress Broadcast)
    {
        byte[] NetAdress_Bytes = NetAdress.GetAddressBytes();
        byte[] Broadcast_Bytes = Broadcast.GetAddressBytes();
        int length = Broadcast_Bytes.Length;
        byte[] Wildcard = new byte[length];
        for (int i = 0; i < length; i++)
        {
            Wildcard[i] = (byte)(Broadcast_Bytes[i] - NetAdress_Bytes[i]);
        }
        return new IPAddress(Wildcard);
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        // Обнаружение адаптеров
        IEnumerable<NetworkInterface> nics = NetworkInterface.GetAllNetworkInterfaces().Where(network =>
(network.NetworkInterfaceType == NetworkInterfaceType.Ethernet || network.NetworkInterfaceType ==
NetworkInterfaceType.Wireless80211));

        // Добавление элементов в раскрывающийся список
        cmbAdptors.DisplayMember = "Description";
        cmbAdptors.ValueMember = "Id";
        foreach (NetworkInterface item in nics)
        {
            cmbAdptors.Items.Add(item);
        }

        if (cmbAdptors.Items.Count > 0)
```

```
            cmbAdptors.SelectedIndex = 0;
        }

private void cmbAdptors_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cmbAdptors.SelectedItem is NetworkInterface)
            {
                slectedNic = cmbAdptors.SelectedItem as NetworkInterface;

 uniCastIPInfo = null;

                ///Заполнение IPv4-адреса
                if (slectedNic != null && slectedNic.GetIPProperties().UnicastAddresses != null)
                {
                    UnicastIPAddressInformationCollection ipInfo = slectedNic.GetIPProperties().UnicastAddresses;

                    foreach (UnicastIPAddressInformation item in ipInfo)
                    {
                        if (item.Address.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork)
                        {
                            //ip adrees
                            IPAddress ip = item.Address;
                            lblIP.Text = ip.ToString();

                            //mask
                            IPAddress mask = GetSubnetMask(ip);
                            //lblNetmask.Text = mask.ToString();


                            uniCastIPInfo = item;
                            break;
                        }
                    }
                }


                BandwidthCalculator(uniCastIPInfo, slectedNic);
                Updator.Enabled = true;
            }
        }

        public void BandwidthCalculator(UnicastIPAddressInformation ipInfo, NetworkInterface selecNic)
        {

            if (selecNic == null)
                return;

            lblType.Text = selecNic.NetworkInterfaceType.ToString();
            lblOp.Text = selecNic.OperationalStatus.ToString();

            //MAC
            PhysicalAddress macData = selecNic.GetPhysicalAddress();
            lblMAC.Text = macData.ToString();


            IPv4InterfaceStatistics interfaceData = selecNic.GetIPv4Statistics();
            int bytesRecivedSpeedValue = (int)(interfaceData.BytesReceived - double.Parse(lblReceived.Text)) / 1024;
```

```
            lblReceived.Text = interfaceData.BytesReceived.ToString();
        lblSpeed.Text = bytesRecivedSpeedValue.ToString() + " KB/s";

        //для графика
        Speed_chart_val = bytesRecivedSpeedValue;


        //Ping
        if (lblOp.Text == "Up")

            {
            string ping_adress = "8.8.8.8";
            Ping ping_sender = new Ping();
            PingReply ping_reply = ping_sender.Send(ping_adress);
            lblPing.Text = ping_reply.RoundtripTime.ToString() + " ms";
            //для графика
            Ping_chart_val = ping_reply.RoundtripTime;
        }
        else
        {
            lblPing.Text = "0";
            //для графика
            Ping_chart_val = 0;
        }


        //Packet loss
        int amount_packet = 15;
        string ip_adress = "127.0.0.1";
        lblLoss.Text = get_loss(ip_adress, amount_packet).ToString() + "%";


        if (ipInfo != null)
        {
            TimeSpan addressLifeTime = TimeSpan.FromSeconds(ipInfo.AddressValidLifetime);
            lblVallifetime.Text = string.Format("{0:D2}h:{1:D2}m:{2:D2}s:{3:D3}ms",
                            addressLifeTime.Hours,
                            addressLifeTime.Minutes,
                            addressLifeTime.Seconds,
                            addressLifeTime.Milliseconds);

        }
    }


      // кнопка для старта графика
    private void button1_Click(object sender, EventArgs e)
    {
        pingThread = new Thread(new ThreadStart(this.getPerformanceCounters_ping));
        pingThread.IsBackground = true;
        pingThread.Start();

        speedThread = new Thread(new ThreadStart(this.getPerformanceCounters_speed));
        speedThread.IsBackground = true;
        speedThread.Start();

        Build = true;
    }

    // кнопка для остановки графика
    private void buttonStop_Click(object sender, EventArgs e)
    {
```

```
            Build = false;
        }

// кнопка для старта расчета
        private void button_calculate_Click(object sender, EventArgs e)
        {
            try
            {
                int int_mask = Convert.ToInt32(textBoxMask.Text);

                        textBoxMask.Text = Get_Mask(textBoxMask.Text);
                lblException.Text = "";
            }
            catch(Exception ex)
            {
                string str_ex = ex.ToString();
                RegexOptions Options = RegexOptions.None;
                Regex regex_ = new Regex("[ ]{2,}", Options);
                str_ex = regex_.Replace(str_ex, "+");
                string exp = str_ex.Remove(str_ex.IndexOf("+"));
                lblException.Text = exp.ToString();
            }

            try
            {
                IPAddress ipaddress = IPAddress.Parse(textBoxIP.Text);
                IPAddress Mask = IPAddress.Parse(textBoxMask.Text);

                //Network address
                IPAddress net_address = Get_Network_Broadcast_Address(ipaddress, Mask, true);
                lblNet_address.Text = net_address.ToString();

                //Broadcast address
                IPAddress broadcast_address = Get_Network_Broadcast_Address(ipaddress, Mask, false);
                lblBroadcast.Text = broadcast_address.ToString();

                //Wildcard mask
                IPAddress Wildcard_mask = Get_Wilcard(net_address, broadcast_address);
                lblbWildcard.Text = Wildcard_mask.ToString();

                //pool
                IPAddress HostMin = Get_ip_pool(net_address, broadcast_address, true);
                IPAddress HostMax = Get_ip_pool(net_address, broadcast_address, false);
                string ip_pool = HostMin.ToString() + "-" + HostMax.ToString();
                lblPool.Text = ip_pool;

                //Hosts
                int amount_ip = Get_Hosts(HostMin, HostMax);
                lblHosts.Text = amount_ip.ToString();


                /////Занятые ip
                ProcessStartInfo arp_a = new ProcessStartInfo("arp", " -a");
                arp_a.UseShellExecute = false;
                arp_a.RedirectStandardOutput = true;
                arp_a.CreateNoWindow = true;
                var proc = Process.Start(arp_a);

                ////строка с результатом
                string s = proc.StandardOutput.ReadToEnd();
```

```
                RegexOptions options = RegexOptions.None;
            Regex regex = new Regex("[ ]{2,}", options);
            s = regex.Replace(s, " ");

            string[] subs = s.Split(' ');
            string usedIp = "";

            for (int i = 10; i < subs.Length; i += 3)
            {

                usedIp = usedIp + subs[i] + "\n";
            }

            lblUsed.Text = usedIp;

            ////////////////////


            lblException.Text = "";
        }
        catch (Exception ex)
        {
            string str_ex = ex.ToString();
            RegexOptions Options = RegexOptions.None;
            Regex regex_ = new Regex("[ ]{2,}", Options);
            str_ex = regex_.Replace(str_ex, "+");
            string exp = str_ex.Remove(str_ex.IndexOf("+"));
            lblException.Text = exp.ToString();
        }

    }


    private void wirelssUpdator_Tick(object sender, EventArgs e)
    {
        BandwidthCalculator(uniCastIPInfo, slectedNic);
    }

  }
}
```

Програмна реалізація файлу Form1.Designer.cs

```
namespace NetworkMonitoring
{
   partial class Form1
   {

      private System.ComponentModel.IContainer components = null;


      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code
private void InitializeComponent()
      {
         this.components = new System.ComponentModel.Container();
         System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
         System.Windows.Forms.DataVisualization.Charting.Series series1 = new
System.Windows.Forms.DataVisualization.Charting.Series();
         System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea2 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
         System.Windows.Forms.DataVisualization.Charting.Series series2 = new
System.Windows.Forms.DataVisualization.Charting.Series();
         this.pingChart = new System.Windows.Forms.DataVisualization.Charting.Chart();
         this.speedChart = new System.Windows.Forms.DataVisualization.Charting.Chart();
         this.button1 = new System.Windows.Forms.Button();
         this.buttonStop = new System.Windows.Forms.Button();
         this.Ping_Chart = new System.Windows.Forms.Label();
         this.Speed_Chart = new System.Windows.Forms.Label();
         this.label1 = new System.Windows.Forms.Label();
         this.cmbAdptors = new System.Windows.Forms.ComboBox();
         this.label2 = new System.Windows.Forms.Label();
         this.lblType = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.lblIP = new System.Windows.Forms.Label();
         this.label4 = new System.Windows.Forms.Label();
         this.lblMAC = new System.Windows.Forms.Label();
         this.label5 = new System.Windows.Forms.Label();
         this.lblOp = new System.Windows.Forms.Label();
         this.label6 = new System.Windows.Forms.Label();
         this.lblVallifetime = new System.Windows.Forms.Label();
         this.label7 = new System.Windows.Forms.Label();
         this.lblSpeed = new System.Windows.Forms.Label();
         this.label8 = new System.Windows.Forms.Label();
         this.lblReceived = new System.Windows.Forms.Label();
         this.label9 = new System.Windows.Forms.Label();
         this.lblPing = new System.Windows.Forms.Label();
         this.label10 = new System.Windows.Forms.Label();
         this.lblLoss = new System.Windows.Forms.Label();
         this.label12 = new System.Windows.Forms.Label();
         this.lblNet_address = new System.Windows.Forms.Label();
         this.label13 = new System.Windows.Forms.Label();
```

```
this.lblBroadcast = new System.Windows.Forms.Label();
this.label11 = new System.Windows.Forms.Label();
this.lblbWildcard = new System.Windows.Forms.Label();
this.label14 = new System.Windows.Forms.Label();
this.lblPool = new System.Windows.Forms.Label();
this.label15 = new System.Windows.Forms.Label();
this.lblHosts = new System.Windows.Forms.Label();
this.label16 = new System.Windows.Forms.Label();
this.lblUsed = new System.Windows.Forms.Label();
this.Updator = new System.Windows.Forms.Timer(this.components);
this.EnterIP = new System.Windows.Forms.Label();
this.textBoxIP = new System.Windows.Forms.MaskedTextBox();
this.EnterMask = new System.Windows.Forms.Label();
this.textBoxMask = new System.Windows.Forms.MaskedTextBox();
this.button_calculate = new System.Windows.Forms.Button();
this.label17 = new System.Windows.Forms.Label();
this.lblException = new System.Windows.Forms.Label();
((System.ComponentModel.ISupportInitialize)(this.pingChart)).BeginInit();
((System.ComponentModel.ISupportInitialize)(this.speedChart)).BeginInit();
this.SuspendLayout();

//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(1907, 889);
this.Controls.Add(this.button1);
this.Controls.Add(this.buttonStop);
this.Controls.Add(this.speedChart);
this.Controls.Add(this.Speed_Chart);
this.Controls.Add(this.pingChart);
this.Controls.Add(this.Ping_Chart);
this.Controls.Add(this.label1);
this.Controls.Add(this.cmbAdptors);
this.Controls.Add(this.label2);
this.Controls.Add(this.lblType);
this.Controls.Add(this.label3);
this.Controls.Add(this.lblIP);
this.Controls.Add(this.label4);
this.Controls.Add(this.lblMAC);
this.Controls.Add(this.label5);
this.Controls.Add(this.lblOp);
this.Controls.Add(this.label6);
this.Controls.Add(this.lblVallifetime);
this.Controls.Add(this.label7);
this.Controls.Add(this.lblSpeed);
this.Controls.Add(this.label8);
this.Controls.Add(this.lblReceived);
this.Controls.Add(this.label9);
this.Controls.Add(this.lblPing);
this.Controls.Add(this.label10);
this.Controls.Add(this.lblLoss);
this.Controls.Add(this.label12);
this.Controls.Add(this.lblNet_address);
this.Controls.Add(this.label13);
this.Controls.Add(this.lblBroadcast);
this.Controls.Add(this.label11);
this.Controls.Add(this.lblbWildcard);
this.Controls.Add(this.label14);
this.Controls.Add(this.lblPool);
this.Controls.Add(this.label15);
```

```
            this.Controls.Add(this.lblHosts);
        this.Controls.Add(this.label16);
        this.Controls.Add(this.lblUsed);
        this.Controls.Add(this.EnterIP);
        this.Controls.Add(this.textBoxIP);
        this.Controls.Add(this.EnterMask);
        this.Controls.Add(this.textBoxMask);
        this.Controls.Add(this.button_calculate);
        this.Controls.Add(this.label17);
        this.Controls.Add(this.lblException);
        this.Margin = new System.Windows.Forms.Padding(4);
        this.Name = "Form1";
        this.Text = "Network monitoring";
        this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
            this.Load += new System.EventHandler(this.Form1_Load);
        ((System.ComponentModel.ISupportInitialize)(this.pingChart)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.speedChart)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();

    }

        #endregion
    //1
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.ComboBox cmbAdptors;
    //2
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label lblType;
    //3
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label lblIP;
    //4
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Label lblMAC;
    //5
    private System.Windows.Forms.Label label5;
    private System.Windows.Forms.Label lblOp;
    //6
    private System.Windows.Forms.Label label6;
    private System.Windows.Forms.Label lblVallifetime;
    //7
    private System.Windows.Forms.Label label7;
    private System.Windows.Forms.Label lblSpeed;
    //8
    private System.Windows.Forms.Label label8;
    private System.Windows.Forms.Label lblReceived;
    //9
    private System.Windows.Forms.Label label9;
    private System.Windows.Forms.Label lblPing;
    //10
    private System.Windows.Forms.Label label10;
    private System.Windows.Forms.Label lblLoss;
    //12
    private System.Windows.Forms.Label label12;
    private System.Windows.Forms.Label lblNet_address;
    //13
    private System.Windows.Forms.Label label13;
    private System.Windows.Forms.Label lblBroadcast;
    //11
        private System.Windows.Forms.Label label11;
    private System.Windows.Forms.Label lblbWildcard;
    //14
```

```
        private System.Windows.Forms.Label label14;
        private System.Windows.Forms.Label lblPool;
        //15
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.Label lblHosts;
        //16
        private System.Windows.Forms.Label label16;
        private System.Windows.Forms.Label lblUsed;

        private System.Windows.Forms.Timer Updator;

        //кнопка для графиков
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button buttonStop;
        //графики
        private System.Windows.Forms.Label Ping_Chart;
        private System.Windows.Forms.DataVisualization.Charting.Chart pingChart;
        private System.Windows.Forms.Label Speed_Chart;
        private System.Windows.Forms.DataVisualization.Charting.Chart speedChart;
        // Enter IP
        private System.Windows.Forms.Label EnterIP;
        private System.Windows.Forms.MaskedTextBox textBoxIP;
        // Enter mask
        private System.Windows.Forms.Label EnterMask;
        private System.Windows.Forms.MaskedTextBox textBoxMask;
        // кнопка для расчета бродкаста....
        private System.Windows.Forms.Button button_calculate;
        // Exception
        private System.Windows.Forms.Label label17;
        private System.Windows.Forms.Label lblException;
    }
}
```